

WHAT'S NEW IN COLDFUSION 4.5

U.S. \$8.99 (Canada \$9.99)

COLD FUSION Developer's Journal

ColdFusionJournal.com

April 2000 Volume: 2 Issue: 4

Announcing...
Coming
July 25-28, 2000 **XML DevCon 2000**
September 24-27, 2000 **JavaCON 2000**

Editorial

Finding a ColdFusion Host

Robert Diamond page 5

Journeyman
ColdFusion

Calling All Custom Tags

Charles Arehart page 20

Foundations

Fusedoc Redux

Hal Helms page 44

CFDJ Marketplace
page 48

CFDJ NEWS
page 50

Allaire Developer
Conference 2000

November 5-8
Washington, DC

Marriott
Wardman Park
Hotel

**SYS-CON
MEDIA**



CFDJ Feature: ColdFusion in the Palm of Your Hand

A developer's guide to creating applications for the wireless world of Palm Pilots

Paul J. Elisii

In Defense of MS Access

You can use Access as the database for CF Web sites, but first adjust your settings

Bruce Van Horn

<BF> on <CF>: ColdFusion & Java - A Match Made in Heaven

Extending CF with server-side Java

Ben Forta

CF & Java: An Online Ticket Store

CF and Java technologies make a good combination

Ajit Sagar

Creating Custom CF Tags as DHTML Wrappers

Turn complex cross-browser DHTML into clean custom tags

Tim Buntel

CFDJ Special Feature: ColdFusion Forms Part 1

A book excerpt on using and creating forms

Ben Forta

Able Solutions

www.ablecommerce.com

Computerjobs.com

www.computerjobs.com

Digital Nation

www.dedicatedsever.com

STEVEN D. DRUCKER, JIM ESTEN, BEN FORTA,
STEVE NELSON, RICHARD SCHULZE, PAUL UNDERWOOD

EDITOR-IN-CHIEF ROBERT DIAMOND
ART DIRECTOR JIM MORGAN
EXECUTIVE EDITOR M'LOU PINKHAM
SENIOR EDITOR JEREMY GELAN
PRODUCTION EDITOR CHERYL VAN SISE
ASSOCIATE EDITOR NANCY VALENTINE
PRODUCT REVIEW EDITOR TOM TAUILLI
TIPS & TECHNIQUES EDITOR MATT NEWBERRY

WRITERS IN THIS ISSUE

CHARLES AREHART, TIM BUNTEL, ROBERT DIAMOND,
PAUL J. ELISSI, BEN FORTA, HAL HELMS,
AJIT SAGAR, BRUCE VAN HORN

SUBSCRIPTIONS

SUBSCRIBE@SYS-CON.COM
FOR SUBSCRIPTIONS AND REQUESTS FOR BULK ORDERS,
PLEASE SEND YOUR LETTERS TO
SUBSCRIPTION DEPARTMENT.

SUBSCRIPTION HOTLINE 800 513-7111
COVER PRICE \$8.99/ISSUE
DOMESTIC \$79/YR. (12 ISSUES)
CANADA/MEXICO \$99/YR
OVERSEAS \$129/YR
BACK ISSUES \$12 EACH

PUBLISHER, PRESIDENT AND CEO UAT A. KIRCAALI
VICE PRESIDENT, PRODUCTION JIM MORGAN
VICE PRESIDENT, MARKETING CARMEN GONZALEZ
ACCOUNTING MANAGER ELI HOROWITZ
ADVERTISING ACCOUNT MANAGER ROBYN FORMA
ADVERTISING ASSISTANT MEGAN RING
ADVERTISING ASSISTANT CHRISTINE RUSSELL
GRAPHIC DESIGNER ALEX BOTERO
GRAPHIC DESIGNER JASON KREMKAU
GRAPHIC DESIGNER ABRAHAM ABBO
GRAPHIC DESIGN INTERN AARATHI VENKATARAMAN
WEBMASTER BRUNO Y. DECAUDIN
WEB EDITOR BARD DEMA
WEB SERVICES INTERDIGITANT B. DAVE
CUSTOMER SERVICE MANAGER CAROL KILDUFF
JDJ STORE.COM JACLYN REDMOND
ONLINE CUSTOMER SERVICE AMANDA MOSKOWITZ
CUSTOMER SERVICE ANN MARIE MILILLO

EDITORIAL OFFICES

SYS-CON PUBLICATIONS, INC. 39 E. CENTRAL AVE.,
PEARL RIVER, NY 10965
TELEPHONE: 914 735-7300 **FAX:** 914 735-6547

COLDFUSION DEVELOPER'S JOURNAL (ISSN #1523-9101)
IS PUBLISHED MONTHLY (12 TIMES A YEAR)
FOR \$79 BY SYS-CON PUBLICATIONS, INC.,
39 E. CENTRAL AVE., PEARL RIVER, NY 10965-2306.

POSTMASTER

SEND ADDRESS CHANGES TO:
COLDFUSION DEVELOPER'S JOURNAL
SYS-CON PUBLICATIONS, INC.
39 E. CENTRAL AVE., PEARL RIVER, NY 10965-2306

© COPYRIGHT

COPYRIGHT © 2000 BY SYS-CON PUBLICATIONS, INC.
ALL RIGHTS RESERVED. NO PART OF THIS PUBLICATION MAY BE
REPRODUCED OR TRANSMITTED IN ANY FORM OR BY ANY MEANS,
ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPY OR ANY
INFORMATION STORAGE AND RETRIEVAL SYSTEM,
WITHOUT WRITTEN PERMISSION.

FOR PROMOTIONAL REPRINTS, CONTACT REPRINT COORDINATOR.

SYS-CON PUBLICATIONS, INC., RESERVES THE RIGHT TO REVISE,
REUBLISH AND AUTHORIZE ITS READERS TO USE
THE ARTICLES SUBMITTED FOR PUBLICATION.

WORLDWIDE DISTRIBUTION

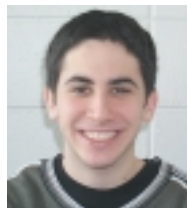
BY **CURTIS CIRCULATION COMPANY** 739 RIVER ROAD,
NEW MILFORD, NJ 07646-3048 PHONE: 201 634-7400

DISTRIBUTED IN USA

BY **INTERNATIONAL PERIODICAL DISTRIBUTORS**
674 VIA DE LA VALLE, SUITE 204, SOLANA BEACH, CA 92075
619 481-5928

ALL BRAND AND PRODUCT NAMES USED ON THESE PAGES
ARE TRADE NAMES, SERVICE MARKS OR TRADEMARKS
OF THEIR RESPECTIVE COMPANIES.

Finding a ColdFusion Host



BY ROBERT DIAMOND

From time to time we receive e-mails from **CFDJ** readers, sent by beginners and advanced developers alike, asking us what we think about a particular hosting company or seeking our opinion as to what a "first-time" developer should be looking for in terms of offered features and pricing, who's the best, who's the cheapest and so on.

At this time there's no one specific company out there that we endorse. From my personal experience and from talking to some of our editors, I can say that there are a number of good ones out there – and a corresponding number of bad ones who'll rip you off on the price, bleed you dry for any additional hosting options, offer little or no support and be down as often as they're up.

There are just far too many hosting companies out there for you to look at them all without the risk of overlooking someone. But thanks to **SYS-CON**, help is on the way.

A convenient place to start is the **ColdFusion Buyer's Guide** and you can find this at www.sys-con.com/coldfusion/BuyersGuide/ or by clicking through **CFDJ's** Web site at www.ColdFusion.com. The Buyer's Guide is a free resource that companies can list themselves in. It's subdivided into listings for 12 key categories and each entry contains both a brief description of the product/services offered and – most important – a link to the Web site of the company concerned.

We currently list a good bunch of hosting companies and more are adding themselves every day, which makes it an ideal place to start looking since it offers you a good overview of the industry. Also, as a quick plug for any hosting companies that might be reading this, listing your product takes less than a minute and can be done at any time.

So let's imagine that you've been to the site, have read a bunch of descriptions, checked out various companies' Web sites, printed out long lists of fea-

tures and looked at the bottom line and highlighted all of the prices....

Whew, possibly after going through all that you took two aspirin and decided to take a quick nap. But where do you go next? Just how good are these companies?

Well, the next place that you should head over to is the **CFDJ Readers Choice Awards** located at www.sys-con.com/coldfusion/readers-choice2000. Voting is possible right up until just a few weeks before the Allaire Conference in November and there are live stats available on the Web site, presenting the most important opinions of all: those of our subscribers, the people who are actually using the various hosting companies.

By now you should have your list narrowed down to just a few choices based on your budget, the features offered and, of course, your own particular needs. But you still may have further questions. With sometimes-pricey startup fees, many discounts for long-term contracts, and the hassle and possible downtime of having to switch companies later on, you really want to make the best choice here and now.

For specific questions, the **ColdFusion Forums** represent another good resource that we offer – it's where writers and readers come together in a community well able to answer your questions. So if you want to know (for example) whether www.BestCFHostingEver.com really is as good as it seems, or if you have nagging reservations about how fast they are so far as support is concerned, then come post your question. Most likely you'll receive several answers.

The flip side to this whole discussion is to consider doing the hosting yourself, which is something that I'll be taking a look at next month, and weighing the pros and cons. Stay tuned!



Robert Diamond

ABOUT THE
AUTHOR
*Robert Diamond is
editor-in-chief of
ColdFusion Developer's
Journal.*

ROBERT@SYS-CON.COM

ColdFusion in the Palm of Your Hand

BY PAUL J. ELISII

A developer's guide to creating CF applications for the wonderful wireless world of Palm Pilots

I am completely hooked on my new wireless Palm! The Palm Pilot VII has changed the way I access information – I can find it anywhere and any time I need it. With the newest Palm I can check account balances in real time on DLJDirect, buy a book from Amazon.com and – using the popular Starbucks Finder application – find the nearest Starbucks. With so many useful wireless applications available, you might be wondering why you can't create your own with ColdFusion. Well, you can! And after reading this article, you'll be well on the way to writing your own wireless applications for the Palm Pilot VII.

Wireless Applications Background

I'm not the only one who's in love with the idea of wireless access to the Web. In a recent study it was reported that the market for wireless handheld devices could

reach 500 million by 2003, which will create a huge demand for Web applications that are "unplugged" and optimized for wireless devices.

For developing wireless applications, two similar but competing protocols for wireless Web access have emerged. The first is the Palm Wireless Web Clipping Protocol – a proprietary but popular solution developed by 3COM. The second is the Wireless Application Protocol. WAP is a more open protocol developed by companies like Unwired Planet, Motorola, Nokia and Ericsson to provide wireless Web access to cellular phones. For this article I'll focus on writing wireless Web applications in ColdFusion for the Palm Pilot Web clipping protocol, and in a future article I'll discuss how to write ColdFusion applications for WAP. Even though these two protocols are different in their implementations, developing applications for each is quite similar.

The palm.net Service

The Palm Pilot VII is a wireless device that contains its own built-in antennae for wireless access to the Web. Once you bring your Palm home, it's easy to sign on and start surfing wirelessly on the Web – simply flip up your Palm's antennae and access the Activate Application to sign on. All it takes is a credit card and about 15 minutes to get started.

3COM charges an online access fee to access the palm.net service. This is a monthly service fee similar to what you'd pay an ISP for Internet access. The palm.net service is a bit pricey compared to your basic ISP costs, and also comes with limits on the number of bytes that can be accessed on a monthly basis. 3COM has basic, extended and volume plans that range from \$10 per month to \$40 per month, with each providing a greater number of KB per month. I pur-

chased the volume plan and, in the first several weeks with fairly active use, I haven't used up even half of my quota.

3COM provides a Web site, www.palm.net, with a lot of great information about the service and a wide array of Web clipping applications that you can download. Available applications include Financial, News, Reference, Travel, Entertainment, Communication, Shopping and Enterprise applications. It also includes a Web browsing function that allows you to access any URL, many stock quote and online trading systems, an application to check airline flight times, and various shopping apps, including Amazon.

Web Clipping Architecture

At the heart of the palm.net service is 3COM's Web clipping proxy server. This server is responsible for converting the standard Internet protocols and content from normal HTML Web pages into a form that's optimized for low-bandwidth wireless transmission and for display on a small device. Figure 1 shows how the proxy server takes the HTTP, SSL and TCP standard protocols and compresses them for wireless access via the Palm Pilot VII.

ColdFusion applications work in this environment just like any other Web page. URL requests are made from the Palm. They go through the wireless network and the proxy to the ultimate destination of the Web server. It doesn't matter that the requested page is an HTML, a CFM or an ASP. As long as the actual content returned from the server is in HTML format that can be processed, compressed and sent through the wireless network, it should work just fine.

Does this mean you can just take your existing ColdFusion apps and make them accessible to the wireless network and they'll work? Well, not quite. As I'll explain in the following section, there are many considerations you have to keep in mind when developing unplugged (wireless) ColdFusion Applications.

The following steps demonstrate the flow of a Web clipping application when launched:

1. If the linked page or graphic is installed on the Palm VII organizer, the page or graphic is displayed without

downloading anything remotely.

2. In other cases the Palm VII organizer compresses the query into a small file in Palm query format and sends this packet over the air to a local base station.
3. The local base station relays the compressed packet to the Web clipping proxy servers on the palm.net service.
4. The Web clipping proxy server translates the query packet into a standard HTTP packet and then sends the decompressed query to the appropriate Internet address. All communication between the Web clipping proxy server and the Internet server is via the standard Internet protocols of TCP, HTTP and SSL.
5. The HTTP server returns an HTML clipping to the Web clipping proxy server.
6. The Web clipping proxy server compresses the HTML clipping into a Palm proxy format file and sends it back to the local base station.
7. The local base station relays the clipping to the Palm VII organizer, where the Web clipping application renders the page for viewing. Communication between the Palm VII organizer and the Web clipping proxy server is via the User Datagram Protocol (UDP).

The Basics of a Palm Web Clipping App

The basic mechanics of authoring Web clipping applications are similar to the mechanics of authoring Web pages in general. However, because of the constraints of authoring for a handheld organizer connected to the Internet via a relatively expensive radio link, you'll need to rethink your philosophy of design and

implementation. The world looks much different when you're holding a small, low-power computer like the Palm VII organizer with its tiny screen, battery-powered operation and relatively slow and expensive wireless connection to the Internet.

Palm applications are called PQAs (Palm query applications). A Web clipping application is a PQA that is developed to take advantage of the wireless capability of the Palm VII.

The basic steps involved in writing a PQA application include the following:

- Produce HTML pages and graphics to serve as the base for the PQA.
- Convert the HTML pages and graphics using a tool called the Query Application Builder to produce the PQA file (to be reviewed later).
- Test the PQA, using either the Palm VII organizer itself or a Palm Emulator, which allows you to run the PQA apps on your workstation.
- Distribute the compiled file. Users install it with HotSync software on their Palm VII organizers.

When developing Web clipping applications, follow the same rules as above. What makes a PQA a Web clipping application is that it has links to pages that are not stored on the Palm but are accessed remotely over the Web. These remote links are coded just like any other Web application link.

Writing a Simple Web Clipping App

The following example shows how you can create the most basic Web clipping application for the Palm Pilot. It's very simple. You can start out with the code in

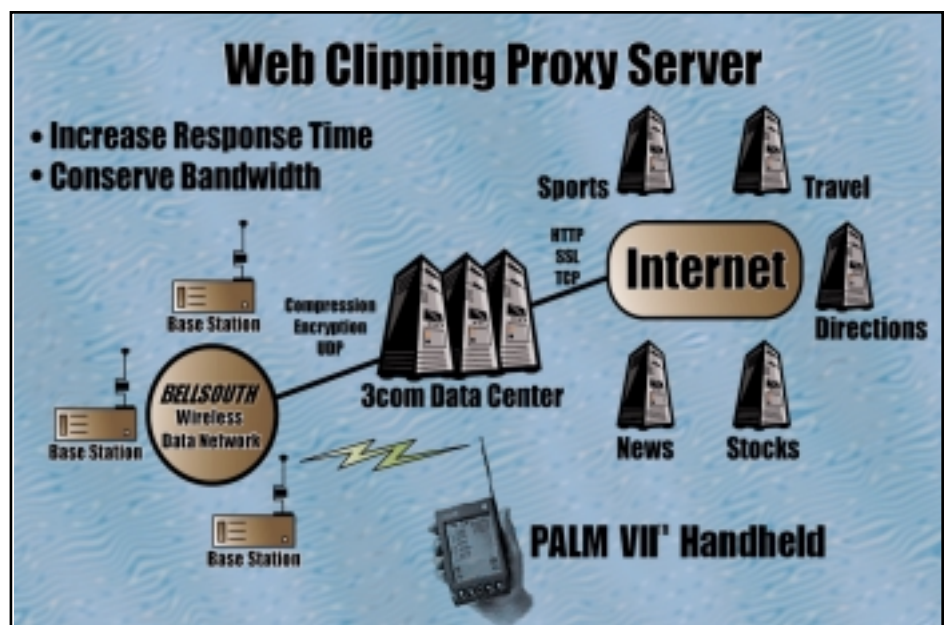


FIGURE 1 Palm VII's Web Clipping architecture

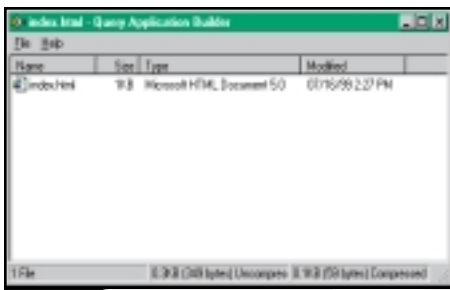


FIGURE 2 Query Application Builder selected files

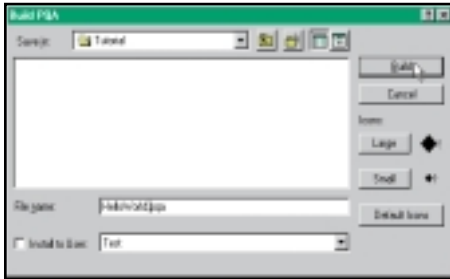


FIGURE 3 Build PQA save options

a single HTML file that does nothing more than display “Hello World.” The entire HTML source code that’s used to build the application is in Listing 1.

The first step in turning the HTML file into a Web clipping app is to create a PQA file through 3COM’s Query Application Builder, which can be found at www.palm.com/dev-zone. (Note that the domain for this is palm.com, not palm.net.)

The Query Application Builder is a simple application that converts your HTML and image files into a single PQA file. Just select the File and Open Index menu to select your HTML file.

Once an HTML file is selected, it’s displayed in the Query Application Builder file window. If the HTML file referenced any local images, they’d be displayed here, as well as included in the PQA file (see Figure 2).

Once the HTML file has been selected, choose “Build PQA” from the File menu. Then the Build PQA Dialog Box specifies the name of the PQA that is being created (see Figure 3).

Note: The icons for this application default to the diamonds with the remote signal displayed on the right in Figure 3. For this application the defaults are being used. In the next example, using ColdFusion, an Allaire image is used for the application icon.

Now that you’ve created the “Hello World” Web clipping application, you can install it to your Palm VII handheld using the Palm desktop software. This is the same process you’d go through to install any PQA application onto your Palm. Just add applications to the desktop software running on your workstation and then load the application to your Palm by synchronizing Palm’s HotSync utility.

Once you add and synchronize the new

PQA, you’ll see the new application appear on your Palm. Now we have a new application icon on our Palm called “Hello World” (see Figure 4).

Clicking on the new “Hello World” application icon will launch our simple application.

Figure 5 shows our simple application running on the Palm. Remember that so far we’ve only accessed local data. Next we’ll get into accessing data over the Web.

Writing a ColdFusion Web Clipping App

In order to write a ColdFusion application for the Palm, we have to involve more than just local pages. As in the previous example, we must add Web clipping pages remotely over the Web.

The two main components in a Web clipping application are as follows:

- **Web clipping application (also known as the PQA):** This is an application you build using HTML, which is installed by the end user onto a handheld organizer. A Web clipping application is like a mini-Web site that’s stored locally, so access to it is instant and free. The index page of the Web clipping application usually contains either a form or a list of links, which are the gateways to the live data provided by your server.
- **Results pages (clippings):** Pages are returned by your back-end server after receiving a request from your Web clipping application. It’s the key part of the overall application, and where all the real work gets done. These pages can be written in HTML, ColdFusion, ASP or whatever. The pages can pull in remote database information and behave similarly to browser-based ColdFusion pages. Images can either be referenced remotely or locally if they were pulled into the PQA. Referring to local images is very useful in saving precious bandwidth, but it leaves less room later to change those images without redistributing the PQA.

The Web clipping pages developed as part of the application can also be accessed through a browser in addition to being accessible through the Palm. Does this mean you can develop an application and deploy it to both the browser and Palm? Not really. Since the Palm only supports a subset of things that can be done on a browser, the UI needs to be specific to the Palm.

One strategy in deploying applications to both the Palm and browser is to separate the user interface from the business and database



FIGURE 4 Hello World application icon



FIGURE 5 Hello World application running



FIGURE 6 ColdFusion application icon

Ektron

www.ektron.com

logic as much as possible so that business logic pages can be included in both the Palm and browser versions of the application. In many ways you can just treat the Palm as a very, very limited browser. The limited nature of this handheld device is equally true with the WAP applications.

Web Clipping HTML Specifics

Web clipping applications are written in a subset of the HTML 3.2 specification. Many of the usual tags for things like tables, forms, checkboxes, radio buttons, select lists and font sizes are supported. However, Web clipping applications don't support more resource-intensive features such as JavaScript, nested tables, frames, cookies or Java.

Web clipping applications are designed to be small and to give users the information they need with minimum effort and maximum speed. Likewise, results coming from your server to a handheld device are designed to be small and to the point. Remember, this is not Web browsing, but Web clipping. Web clipping applications should be written to provide users only with the information they need, without any extraneous items. Try limiting the size of your result pages to about 400 bytes.

There are two meta tags you'll want to consider incorporating into your Web clipping applications:

- **PalmComputingPlatform:** Identifies your page as a "Palm Friendly" page – i.e., one that's been designed to work well on a small screen and isn't wasteful of bandwidth. When you include this tag, images will be rendered and the entire text will be displayed on the device. Otherwise, images will be stripped out and only the first 1,024 bytes of your page will display. This meta tag should be used on all of your local and remote clipping pages.

```
<meta name="PalmComputingPlatform" content="true">
```

- **LocalIcon:** Stores graphics and HTML documents not otherwise referenced locally within your Web clipping application. It is to be used only in local pages and should be placed only in the root page of the Web clipping application.

```
<meta name="LocalIcon" content="images/pal_m_image.gif">
```

Two other meta tags are also useful.

- **HistoryListText:** Specifies the user-visible string for each clipping displayed in the history pop-up menu on the Palm. This meta tag can be used on all of your Web clipping pages.

```
<meta name="HistoryListText" content="true">
```

- **PalmLauncherRevision:** Sets the version string for your Web clipping application. To be used on the main index (or root) page of your Web clipping application. This string can be viewed in the Info window of the Applications Launcher. The version in my example below is 1.0, but it can be whatever is meaningful for your application.

```
<meta name="PalmLauncherRevision" content="1.0">
```

Web clipping applications also support secure HTTPS connections to make credit card purchases and stock transactions possible. The Palm indicates a link is an over-the-air secure link by automatically displaying the top icon at right on all anchor tags, except where a user-defined image is used. All other over-the-air links have the bottom icon at right next to the link. To ensure consistency with other Palm applications, user-defined images should include either the secure icon or the nonsecure icon.

In Web clipping applications there are three types of links that you'll be using. The first is a link from one local page to another on the Palm – a remote link to a remote link. The syntax for this link is the following:

```
<a href="about.html">
```

The second type of link is from a local page to a remote page. The syntax for this link is the following:

```
<a href=http://www.allaire.com/about.html>
```

The third type of link is from a remote page to a local page stored on the Palm. The syntax for this link is the following:

```
<a href=file:coldfusion.pqa/about.html">
```

Since the Palm screen is much smaller than a browser page, graphics should be used with care. GIF and JPEG images are both supported by the Web clipping system. However, the usable screen size is only 153 pixels wide by 144 pixels high and only gray-scale images are supported.

The idea in developing Web clipping applications is to keep the images small and to reference them locally wherever possible.

The LocalIcon meta tag is used to store graphics locally, using the following syntax:

```
<meta name="LocalIcon"
content="images/pal_m_image.gif">
```

It can be referred to remotely in a clipping page using the following syntax:

```
<td><IMG height=30
src=coldfusion.pqa/pal_m_image.gif"
width=30></td>
```



FIGURE 7: ColdFusion application initial page



FIGURE 8: ColdFusion application form page



FIGURE 9: ColdFusion application results page

The following application shows how a ColdFusion application interacts with a PQA. The application is a simple ColdFusion application that contains a form that accepts a keyword and then passes it to a result page that queries a database of contacts and returns the results.

Listing 2 shows the code associated with the HTML file used to build the PQA. This form loads a local image to be displayed on the page and contains a link to my ColdFusion page (the Web clipping).

This sample page makes use of a table that contains some text and a local image reference. In addition, it contains the link that references the remote clipping page

Listing 3 shows the source code of the simple ColdFusion page that is loaded via the Web onto the Palm.

This page makes use of an input form with a submit button that sends the form variable to the ColdFusion result page to query and display the results. The resulting page is shown in Figure 8.

Listing 4 shows the ColdFusion code that accepts the form variable from the input page and performs the query to return the results back.

This page makes use of a ColdFusion query tag that selects the list of matching contacts and an output tag that displays the results to the Palm screen. You can see what the page looks like on the Palm in Figure 9.

gives you the ability to roll out applications to handheld devices. This process allows access to those applications for users at any time and in any place. The Palm platform provides a very nice vehicle for deploying wireless applications in ColdFusion.



Portions of this article were taken from texts and articles on developing Web clipping applications from the Palm development zone

Paul J. Elisii is the founder and CEO of e-Tech Solutions, Inc., a Philadelphia-based e-business development company and an Allaire Premier Partner. Among other things, Paul is active in developing and researching emerging technologies for e-Tech Solutions.

Developing wireless Web applications

pelisii@etechsolutions.com

```
<html>
<head>
<title>Hello World! </title>
<meta name="Platform" content="true">
</head>

<body>
Hello World!
</body>
</html>
```

```
<html>
<head>
  <title>ColdFusion!</title>
  <meta name="Pal mComputingPl atform" content="true">
  <meta name="Local I con" content="i mages/al l ai re. gi f">
</head>

<body>

<table>
  <tr>
    <td><IMG height=30 src="i mages/al l ai re. gi f" width=30></TD>
    <td>The fol lowi ng i s a sample Col dFusi on Web Cl ippi ng Appl i ca-
ti on</TD>
  </tr>
</table>

<a href="http://www. etechdev. com/pal m/col dFusi on_ form. cfm">Launch
Col dFusi on Sample Appl </a>

</body>
</html>
```

```
<html>

<head>
  <title>ColdFusion! </title>
  <meta name="Pal mComputingPl atform" content=" true">
</head>

<body>

Enter part of a company name or last name and hit search.
<form action="coldfusion_resul t.cfm" method=post>
  <INPUT name=Keyword>
  <INPUT name=Search type=submit value=Search>
</form>

</body>
</html>
```

```
<html>
<head>
<title>ColdFusion!</title>
<meta name="PalmComputingPlatform" content="true">
</head>

<body>

<!-- Query the database for the available contents by last name and
company. -->
<cfset myvar='% ' & UCASE(form.keyword) & '% ' >
<CFQUERY NAME="contacts" DATASOURCE="Sales">
select c.company_name, p.first_name, p.last_name, p.phone_1,
address_line_1, city,
state_province, zip_code
from company c, people p, address a
where c.client_id = p.client_id and a.client_id = c.client_id and
a.record_type = '1' and
(ucase(company_name) like '#myvar#' or ucase(last_name)
like '#myvar#')
order by company_name, last_name
</CFQUERY>

<!-- Display the records found. -->
<CFOUTPUT QUERY="contacts">
<tr>
<td>
<b>#contacts.company_name#</b><br>
#contacts.first_name# #contacts.last_name#<br>
#contacts.phone_1#<br>
#contacts.address_line_1#<br>
#contacts.city# #contacts.state_province#
#contacts.zip_code#<br><br><br><br>
</td>
</tr>
</CFOUTPUT>

<!-- If no records were found matching keyword, then return message
-->
<cfif contacts.recordcount EQ 0>
No Records Found.
</cfif>

</body>
</html>
```

● ● ● ● ● ● ● ● ● ● ● ● ●

The code listing for
this article can also be located at
www.ColdFusionJournal.com

In Defense of MS Access

You can use Access as the database for CF Web sites, but first adjust your settings

BY
BRUCE
VAN HORN



I often hear ColdFusion developers and some of my colleagues who do training for Allaire say things like, "Don't just walk away from MS Access...run!" While I think I know much of the rationale for a statement like that, I still feel I have to come to its defense.

The Bad News First!

Let me start by saying that I agree, to some extent, with the above derogatory comment about Access. Much of the criticism of ColdFusion comes in the form of "CF is too slow." When asked to explain, people will say, "It seems like most of the ColdFusion sites I visit are slow" or some similar remark about the response time of searches from ColdFusion-driven sites. This sentiment has been a thorn in Allaire's side since the very beginning. The reality is that ColdFusion isn't slow, but MS Access is. Chances are the sites in question are using MS Access as their back-end database. So the problem is really one created by Access, but often it's CF that gets the blame.

Let's face the fact that there are some real shortcomings to using MS Access as the database for our ColdFusion Web sites. The primary issues are speed, concurrency and the amount of data to be stored. According to its own documentation, Access 2000 has a physical file size limitation of 2 gigabytes and cannot support more than 255 concurrent users. In an enterprise-level Web application it doesn't take long to reach 2 gigabytes of data and I'd hate to see the response time if there really were 255 people requesting data from Access simultaneously.

As for speed, there's no question that Access is much slower than enterprise-level RDBMS applications like MS SQL Server or Oracle. (In a benchmark study I wrote a CF template that ran the same query against a table that returns 1,132 records in both Access and SQL Server 7.0. The average execution time – using CFQUERY.ExecutionTime – reported by the SQL Server 7.0 query was 81 milliseconds and the average execution time from the Access query was 381 milliseconds.)

One other limitation is the inability to leverage the power and speed of stored procedures and triggers. Stored procedures are simply precompiled SQL statements that execute much faster because they're already compiled on the server. Triggers are simply stored procedures that can be executed automatically by the server whenever data is changed. For example, a common need is to insert a record into the database and then immediately query the database to retrieve the primary key of the newly inserted record. Rather than write a separate query to retrieve this information, you can create a trigger (see Listing 1) that will automatically return that information any time a record is inserted.

So if you're planning to build a high-traffic Web site with a large amount of data that needs to respond very quickly, don't even think about using MS Access as your production database.

That said, however, there are some very legitimate places to use MS Access as the database for your ColdFusion applications.

Data Model Prototyping

Though the vast majority of CF applications I create for my customers ultimately use MS SQL Server 7.0 or Oracle as their production RDBMS, I do 100% of my initial development using MS Access. The reason is simple: I have yet to find a better tool than MS Access for doing database prototyping! In the early stages of development the data model for the application will need constant molding and refining. Access gives me a very easy environment in which to create, delete and restructure tables, redefine column types and create or modify primary key definitions. It may be nothing

more than my personal opinion, but it just seems easier to open up Access to make changes to a table than it is to do the same thing through SQL Server's Enterprise Manager.

Upsizing to a real RDBMS

I have no qualms about developing my database in Access because it's so easy to upsize to a real RDBMS when the time comes. MS Access 2000 comes with a very easy-to-use Upsizing Wizard (see Figure 1) that walks you through a few simple steps to convert your database to SQL Server 7.0. Even without the wizard, it's not much harder to bring the table structures and the data into Oracle by writing a few SQL scripts.

Portability

Another reason I use Access in the early stages of application development is portability. I may start off by creating the database for an application on my laptop while in a hotel room when I'm on the road, training for Allaire. Then I'll need to share that database with other developers when I get back to the office. It sure is easier to copy a single .mdb file than it is to migrate the data from one server to another.

My clients may introduce another portability issue. A client may want us to create a test site on their server, but may not have a license for Oracle or SQL Server. All I have to do is upload the Access file to their server and create an ODBC Datasource in the CF Administrator that points to it – we're up and running in a matter of minutes and we didn't have to spend any money on additional software.

Low- to Moderate-Traffic Web Sites

As I said earlier, there should be no doubt that Access just isn't suitable for

Watchfire

www.watchfire.com

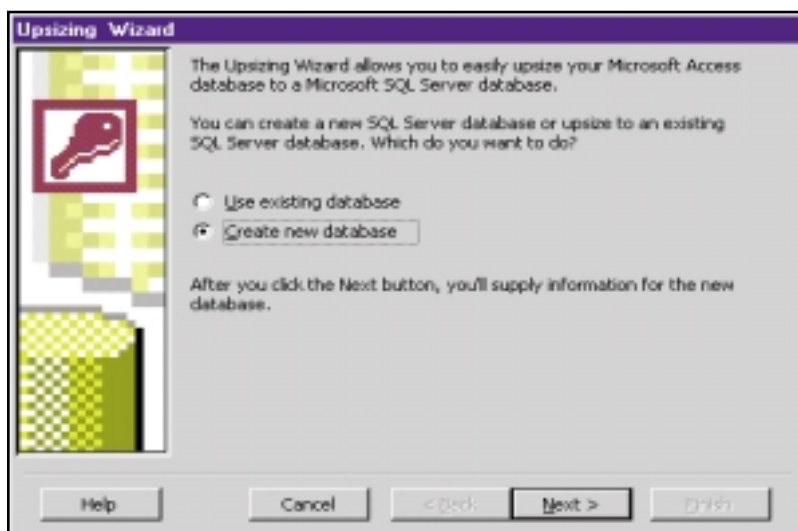


FIGURE 14 MS Access 2000's Upsizing Wizard

prime-time, high-traffic Web sites. There are significant scalability issues that simply cannot be resolved using Access. But let's face the truth: the vast majority of companies that want to set up a data-driven Web site won't be receiving 100,000 or more visitors a day! Most companies and organizations would be ecstatic to receive 5,000 visitors a day. Not everybody who wants a data-driven Web site is a ToysRUs.com or a SmartMoney.com – they all want to be there someday, but they aren't there yet.

For these companies, organizations, groups and individuals that have fairly light traffic levels and response time isn't tremendously important, MS Access is probably going to be adequate to meet the demands of their limited audience. It's a great way to get started without the up-front expenses of buying an RDBMS like Oracle or SQL Server, not to mention the hardware to put those applications on.

Provided that the database is designed correctly and the ColdFusion code is written properly (granted, these are two fairly large assumptions), Access can actually perform quite well under moderate load. As a developer, there are a few steps you can take to ensure that your Access-based application will work better.

Improving the Performance of Your Applications

Here are a few tips that will greatly improve the performance and sta-

bility of your ColdFusion/Access applications.

BLOCKFACTOR

First, add the `BLOCKFACTOR="100"` attribute to all your `CFQUERY` tags (see Listing 2). This alone will dramatically increase the speed of your queries. Without this attribute, when you run a query, ODBC hands the retrieved records back to the CF Server one at a time. By adding this attribute, ODBC will keep the records and then hand them off to CF in blocks of 100 at a time, which is much faster. In a benchmark test I ran against an Access table, the average response time without the use of blockfactor was 420 milliseconds. Just by adding `BLOCKFACTOR="100"` to the `CFQUERY` tag, the average response time dropped to 97.19 milliseconds for the very same query! This is an attribute you can use on any Oracle or ODBC datasource, not just Access. You can even use it on queries that only retrieve a few records – the response time will still be faster than without it.

Cache Your Queries

You can also improve the performance of your applications and the load placed on your database by caching your queries wherever you can. If you have a page that retrieves information from a database and that information is the same for everyone who runs that page, that query is probably a good candidate for query caching. The theory is this: let the first

request for that page load the query into the server's memory. Subsequent requests for the same data can get it much faster from the server's memory than the server can get it from the database. There are basically two ways to cache a query: store it in a variable or use the `CACHEDWITHIN` attribute of the `CFQUERY` tag.

If the data retrieved from the query never or rarely changes, it might make sense to cache that query into an application-level variable. Once the variable has been created, it will stay in the server's memory and be available to all the pages in that application until the application times out (see the ColdFusion Administrator for establishing default timeouts for application and session variables). In this example (see Listing 3) the query creates a variable in the Application scope (notice the `NAME` attribute of the query tag) and the query is nested inside a `CFIF` block to check if the variable already exists. If it doesn't exist, the query runs and creates the variable, so the `CFIF` block will not run the next time the page is loaded. The `CFLOCK` tag ensures that only one user actually creates this query, just in case there are simultaneous requests for this page.

If the data returned by the query changes frequently, but those changes don't have to be seen immediately, you should look into the `CACHEDWITHIN` attribute of the `CFQUERY` tag. Basically, you just need to determine how long the results of that query should be kept in memory before going back to the database to see if anything has changed. You'll want to use the `CreateTimeSpan` function to determine the length of time to cache each query (see Listing 4). The first request for that query will put the results in the server's memory and it will stay there until the time limit is reached – in this example, for 30 minutes.

Using either of the above caching methods will improve the performance of the application and will reduce the amount of hits to your server's hard drives. Caching methods should be considered for all of your ColdFusion applications, not just those that use Access as the database.

Allaire

www.allaire.com

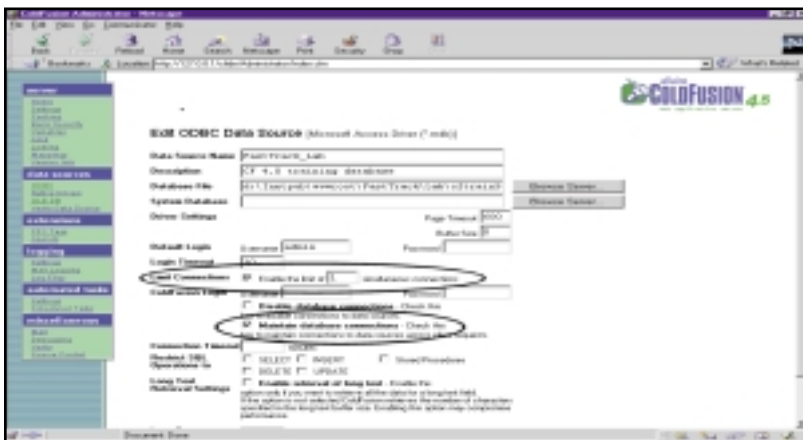


FIGURE 2: Adjusting the settings in ColdFusion Administrator

Limiting Simultaneous Connections and Maintaining Database Connections

If you're using Access as your database on your Web site, you should definitely consider these two settings in the ColdFusion Administrator.

I usually say to my students: "Access is very bad at managing simultaneous connections but ColdFusion is great at simultaneous connections." The idea here is to let CF manage the connections, not Access. When you create an ODBC datasource for an Access database, the ColdFusion Administrator gives you the option of enabling the limit of x simultaneous connections to the database (see Figure 2). I strongly recommend you set that limit to 1. ColdFusion will then allow only one request at a time to be sent to Access,

and it will queue subsequent requests.

The end result is that everyone will get their data faster by letting CF do the managing of requests, not Access. This will also keep your application from crashing during bursts of heavy requests, which is very likely if you let Access handle the concurrency issue. Note that this setting should only be used for Access or other file-based data-sources such as dBase, Paradox and Excel, not for Oracle or SQL Server.

The other setting that you should definitely enable is the “Maintain Database Connection” option (see Figure 2).

Without this setting, CF will establish a connection to the Access database, lock the .mdb file for exclusive use, pass in the request and then

release its connection. It will go through this process with every request for that datasource, which significantly slows the process. By telling CF to maintain the connection, it only has to go through the process for the first request. Subsequent requests are much faster because CF has maintained the connection to the file. Note that this can be rather annoying on a development server, because CF will lock the file, and not allow you to modify the database until it releases the connection.

Summary

Broad statements like “don’t walk away from Access...run” shouldn’t be made without giving some context. While Access has many shortcomings and should not be considered as a back-end database for enterprise-level Web sites, it does have a very valuable contribution to make to the world of data-driven Web sites and to the development of those sites. So, if you’re thinking about abandoning Access as your database just because you heard some rather respected people in the industry badmouth it, think again. It’s a great development tool and can be a more-than-adequate solution to your data-driven Web site, given the right environment.



ABOUT THE AUTHOR

Bruce Van Horn is an Allaire certified instructor and president of Netsite Dynamics.

BRUCE@NETSITEDYNAMICS.COM

LISTING 1

```
<!-- This SQL code (for MS SQL Server 7.0) will create a trigger
that will return the automatically generated primary key (Employee_ID)
any time a new record is inserted into the Employees table -->
```

```
CREATE TRIGGER [GetEmp_PK] ON [Employees]
FOR INSERT
AS
SELECT Employee_ID FROM INSERTED
```

```
<!-- This is how you would use the above trigger in your CF code
Notice that you actually name the CFQUERY tag in this case,
which you normally would not do for an insert operation -->
```

```
<CFQUERY NAME="AddEmpl oyee" DATASOURCE="DHPCCA_SQL">
    INSERT INTO Empl oyees (F irstName, LastName)
        Values ('Bruce', 'Van Horn')
</CFQUERY>
<CFOUTPUT QUERY="AddEmpl oyee">
Empl oyee ID generated:  #AddEmpl oyee. Empl oyee_ID#
</CFOUTPUT>
```

LISTING 2

```
<CFQUERY NAME="GetDist" DATASOURCE="FastTrack_Lab" BLOCKFACTOR="100">
  SELECT Distributor_ID, Distributor_Name, City, State_Region, Phone,
  Email
  FROM Distributor
  ORDER BY Distributor_Name
</CFQUERY>
```

LISTING 3

```
<CFIF NOT IsDefined("Application.GetDist")>
  <CFLOCK SCOPE="Application" TIMEOUT="30" TYPE="Exclusive">
    <CFQUERY NAME="Application.GetDist" DATASOURCE="FastTrack_Lab"
BLOCKFACTOR="100">
      SELECT Distributor_ID, Distributor_Name, City, State_Regions,
Phone, Email
      FROM Distributor
      ORDER BY Distributor_Name
    </CFQUERY>
  </CFLOCK>
</CFIF>
```

LISTING 4

```
<CFQUERY NAME="GetDiSt" DATASOURCE="FastTrack_Lab" BLOCKFACTOR="100"
CACHEDWITHTIME="#CreateTimeSpan(0,0,30,0)#">
  SELECT Distributor_ID, Distributor_Name, City, State_Region, Phone,
Email
  FROM Distributor
  ORDER BY Distributor_Name
</CFQUERY>
```

CODE LISTING

DDDDDDDDDD

The code listing for
this article can also be located at
www.ColdFusionJournal.com

Interact

www.interact.com

ColdFusion and Java—A Match Made in E-Heaven

When you need to extend ColdFusion, take a serious look at server-side Java



BY
BEN
FORTA

Last month I stated that server-side Java was an ideal way to extend ColdFusion, and that Java would be playing a key role in ColdFusion's future. This month I'd like to continue this discussion with an overview of what Java is, what some of those acronyms mean – and what all this has to do with ColdFusion.

Understanding Java

Java started life as yet another programming language, but has quickly evolved to become a complete platform for next-generation computing. Created by Sun, and now supported by many leading software vendors, Java includes component architectures, messaging components, transaction services, security and even APIs for things like wireless communication, telephony integration, smart cards and more.

The Java Language

The original promise of Java was portability – the ability to write code once and have it execute on any platform. Portability is nothing new: C (and C++) promised it originally, and for the most part failed to deliver on that promise.

But there's a significant difference between how C and Java facilitate portability. C code is source code-portable; as long as standard libraries are used, the same source code can (at least in theory) be compiled for multiple platforms. C code can't be executed as is. It needs to be compiled (turned into binary code with instructions the computer can understand). The compiled code is specific to the platform for which it was compiled and thus is not portable. Only the source code is portable.

Java takes a different approach to portability. Instead of requiring that the source be compiled for each platform, Java code is compiled once into bytecode, and that bytecode is portable.

The Java Virtual Machine

So how can Java be portable when compiled while C cannot? The answer is that Java isn't actually run by your operating system. Instead, it runs within a special environment that must be installed on your computer, the Java Virtual Machine.

As its name suggests, the JVM is a virtual computer that runs on your computer. It supports a special instruction set, and features all the basic functionality that an operating system requires (like file I/O). Java applications run inside that virtual machine. The JVM is very machine and operating system specific, and the internal workings differ greatly from one JVM to another. But those differences are hidden inside the JVM itself. The Java bytecode that runs within the JVM needn't be aware of them at all.

Virtual machines can be thought of as interpreters. They create a consistent environment within which applications can run, and they interpret instructions to operating system-specific calls as needed.

To run Java applications, you must have a JVM installed. The JVM is part of the Java Runtime Environment (JRE), which also includes core Java classes and other supporting files.

Beans

Like any good programming language, Java supports reusable components. These are called *beans*, and there is an entire specification that dictates how they are to be used and developed. Put in different terms (and at the risk of offending Java diehards), beans do for Java

what custom tags do for ColdFusion developers.

Enterprise JavaBeans are beans that are part of a scalable, transactional, multiuser-safe, server-side application. EJB run in an EJB server (like JRun 3.0) and are accessed via containers.

Servlets

Servlets are a special form of Java application designed specifically for Web-based applications. They provide simplified access to everything from GET and POST data to cookies and a whole lot more. They can also use beans. Servlets can be used wherever ColdFusion pages are used, as long as you write your code in Java.

To execute servlets, you need an application server that supports them. One of the leading servlet application servers is Allaire's JRun.

Note: What do you need to start playing with servlets and JSP? Download JRun from the Allaire Web site. To learn more about Java in general, visit Sun's Java site at <http://java.sun.com>.

JavaServer Pages

JavaServer Pages is a scripting language used to write servlets. JSP developers use special tags (sound familiar?) and scriptlets (intermingled with HTML and any other client-side technology) to create pages. These pages are compiled into servlets that are then executed to process user requests.

Java 2 Platform, Enterprise Edition

J2EE is Sun's new Java platform for developing enterprise applications on top of Java technology. J2EE is less a technology per se than a collection of Java technologies (including EJB, servlets and JSP) that, when combined, provide a complete enterprise-level application development platform.

Java and ColdFusion

So what has all this got to do with ColdFusion? Right now, not that much, but that's about to change. In addition to ColdFusion, Allaire produces another application server, JRun. JRun is a Java-based server that fully supports EJB, servlets and JSP.

ColdFusion and JRun have their respective strengths. The former dominates rapid development – there's no faster way to develop Web-based applications. ColdFusion also features a rich set of core services (including security, scalability and all sorts of extensions). JRun's big advantage is that it is Java, and thus offers all the robustness, scalability, APIs, components, third-party technologies, developer resources and mass acceptance that Java can lay claim to.

And Allaire has now publicly announced that ColdFusion and JRun are going to be merged into a new killer application server – one single product that offers the best of both worlds. For ColdFusion developers this means improved performance, greater extensibility, a massive array of usable services and technologies, and far more respect from the outside world. For Java developers this means faster development (they'll be able to use CFML) and simpler access to core services that CFers have taken for granted for so long.

It's a win-win situation. Developers who want to write in CFML and not think about Java will be able to write just as they do now in ColdFusion. Similarly, developers who want to write servlets can continue to do what they do now in JRun. But for developers wanting to take their applications to the next level, this new application server (code-named Pharaoh) will open doors to a whole new world of opportunities.

Summary

Java is going to play a key role in ColdFusion's future, and the beginnings of this grand plan are visible in ColdFusion 4.5 with its ability to interact with beans, classes and servlets. And as I said last month, this is why I'd recommend that when you need to extend ColdFusion, take a serious look at server-side Java. Besides being a wonderful technology, it's also going to be part of your life in the not-so-distant future. By jumping on the Java bandwagon now, you'll be on track to take advantage of what Allaire has in store for you.



ABOUT THE AUTHOR

Ben Forta is Allaire Corporation's product evangelist for the ColdFusion product line. He is the author of the best-selling ColdFusion 4.0 Web Application Construction Kit and its sequel, Advanced ColdFusion 4.0 Development (both published by Que), and he recently released Sams Teach Yourself SQL in 10 Minutes.

BEN@FORTA.COM

Inteliant

www.inteliant.com

XML DevCon 2000

www.xmldevcon2000.com

XML DevCon 2000

www.xmldevcon2000.com

An Online Ticket Store

CF and Java technologies make a good combination



BY
AJIT
SAGAR

In the January issue of **CFDJ** we walked through the development of the online "store-front." As a software architect, when designing an enterprise-level application, you frequently have to justify the choices you make in the technologies you select for the solution. You often end up answering a trio of questions – "why?", "what?" and "how?"

This month, accordingly, I'd like to step back from the development of the Online Store application (the "what?" and the "how?") and focus on the "why?" Several readers have asked why using CF template pages with Java servlets is such a good idea. The whole application could have been built using just the CF application server. Similarly, the application could have been built using just Java technologies. These are valid questions. Coincidentally, I've been asked the same questions at my place of work. With the advent of J2EE (Java 2 Enterprise Edition) and the emergence of Java Servlets/JavaServer Pages (JSPs), why would Web developers look at a CF application server and use a "proprietary" markup language for creating business applications? If your company commits to a J2EE architecture, where does CF fit? Does it fit?

But First: Why an Online Ticket Store?

Before discussing the application, I'd like to take a moment to mention how the idea for the application originated. Last year, I worked on a project that involved setting up a product configuration site for some of our core engines. The functionality of these core engines was accessible via a Java API. One alternative for creating the site was to use Java's client-side technologies, that is, applets and Swing components, for creating the browser clients, but there were some problems with this approach. One of the main ones was that, although the functionality could be accessed from a Java/HTML client

through a browser, we didn't have any Web developers who could develop the site in Java. Our computer graphics and media groups were more versed in Allaire's CF, and therefore better suited for creating the site than Java developers from disparate Java groups. After all, that's really part of their job description – but it's not part of the job description for Java developers.

CF allowed us to create the "store-front." The user preferences and accounts were maintained in a database directly accessible via CF tags. All the graphics and interactions for the user interface were handled in CF and JavaScript. For highly interactive pages we leveraged Java applets. However, there was the dilemma of making HTML data available from the Java engines to the CF pages. Java servlets offer the ideal mechanism for serving up HTML to the client. At that time (February 1999), however, there was no published way to access servlets from a CF page. So my colleague and I created a simple string-based protocol to exchange data from a CF page and a Java servlet by passing the parameters on a URL

command line.

As a part of the application, I developed a custom tag for CF allowing us to access servlets from ColdFusion. I wanted to demonstrate how CF and Java technologies could cooperate to create a distributed application, and I wanted to use an example business that would span all the tiers of a distributed application. Hence, the Online Ticket Store was born.

Supplementary Technologies

CF and Java offer supplementary technologies. There's definitely an overlap in some of the functionality, but if the design is chosen carefully, application developers can benefit from the features offered by both Java and CF environments. The CF application server allows a developer to create virtual online storefronts and sites. Java has technologies that cover the entire span of a distributed application, from the user interface to back-end adapters that extract the data from legacy engines. However, creating sophisticated template-based user interfaces and Web sites is not Java's forte. The computer industry has a large segment of developers who are involved in rapid application development of Web-based systems, but they're not necessarily programmers since they don't program using traditional languages such as C++ and Java. Their skills are more suited to Web development using scripting languages.

At the same time, in an enterprise-wide application, there's a need for programmers who can cre-

Seventh in a series of articles focused on using ColdFusion and Java technologies to develop a Ticket Store application. The first article in this series was published in the July/August 1999 issue of CFDJ (Vol. 1, issue 4), and the next four were published in Java Developer's Journal. The sixth installment appeared in January CFDJ (Vol. 2, issue 1).

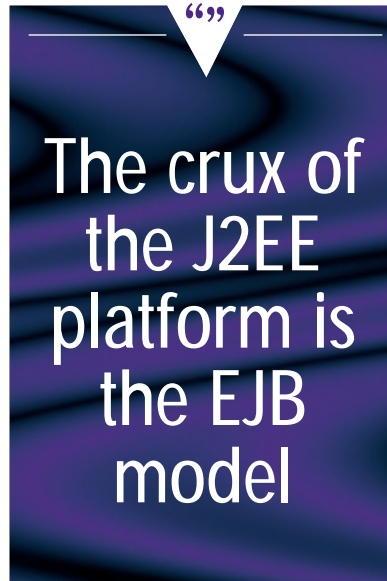
Allaire

www.allaire.com

ate the business logic for these applications and serve it up to the client. They're traditional programmers who program in traditional programming languages.

The Broad Acceptance of HTML UIs

Over the past few years, Java made a bid as a language and platform that can be used to develop enterprise-level distributed applications. While it has grabbed a large part of the territory on the middle tier and the server side, HTML-based clients have clearly won the Internet war. Most of the applications are moving toward pure HTML user interfaces for Web applications. Chances are your customer will require a browser-based interface in addition to other types of clients. Java applets are suitable for some GUI components, but they're not appropriate for developing the entire client. Java servlets, on the other hand, are used to serving up HTML to the browser client from the server. JavaServer Pages



are similar to Active Server Pages (hence the name JSPs) and allow developers to embed programming code inside HTML pages. CFML allows the embedding of ColdFusion tags to make server-side calls. The purpose of these mechanisms is the same – to serve up HTML to the client.

Middleware Access

In order to access server-side business objects, the middleware components of a distributed architecture need to support access protocols for the popular object models – COM, EJB, CORBA and Java objects. In the Java world this access is provided through Java servlets and JSPs. JSPs exist in the form of embedded Java code in the HTML page. JSPs are compiled into servlets the first time they're invoked and thereafter function as normal servlets. The advantage of JSPs is the same as that of ASPs. They can be used by Web developers to create Web presentation components such as shopping carts.

CF allows access to server-side components directly via the WDDX serialization mechanism. These components may be based on Java, COM or CORBA object models. Servlets enable access to Java server components, which may be Java Objects or Enterprise JavaBeans.

Entech

www.entech.com

Isite

www.isitedesign.com

Presentation Vs Business Components

Middleware components may be divided into two broad categories – presentation and business logic. The presentation components are responsible for generating data for the client. Business logic components are responsible for extracting data from the back-office applications. Decoupling business logic components from presentation components has several advantages – reusability, scalability, simplicity of design and so on. Tools such as CF are more suited to building presentation components. Technologies such as Java servlets and JSPs are better for building client-side components. Their combination enables a more robust, scalable and manageable application.

The crux of the J2EE platform is the EJB model. EJBs are business logic components that execute in a Java application server. Servlets and JSPs are a means of accessing EJBs from the Web. Therefore, CF tags can create the presentation components that “execute” in the CF appli-

cation server. These components can access business logic in the form of Java or EJB objects via servlets. The business logic components can provide access to the back-office applications.

Conclusion

I'd like to list some reasons why CF and Java technologies make a good combination:

- From a project management and development point of view, the team that builds the storefront for your application typically consists of people who understand Web development and graphics, but not necessarily Java. Once you have standard templates in JSPs that allow you to access server-side components through plain scripting, maybe you won't need the decoupling. However, my experience has been that the guys writing Java code for the Web development team usually don't understand the issues of Web site and storefront development very well.

- CF allows you to abstract the access mechanism to the objects on the server side. JSPs mean that you are tied to Java. If you want an abstraction layer that allows you to get to other technologies – for example, COM on the server side – CF is a good tool to build that abstraction layer.
- Some components, such as a payment module that's hooked to a credit card service, don't need to connect to the server side and can be handled at the Web front. CF allows easy integration with such systems.
- CF can be used for rapid prototyping. Since it allows access to different data sources via scripting, you can build up a site without a huge amount of investment.
- Due to the hooks into LDAP and so on, you can build a level of role-based personalization using CF.




ABOUT THE AUTHOR

Ajit Sagar is a senior solutions scientist in a firm specializing in B2B marketplaces. A Sun-certified Java programmer with nine years of programming experience, including three in Java, Ajit holds an MS in computer science and a BS in electrical engineering.

AJIT@SYS-CON.COM

VirtualScape

www.virtualscape.com



CREATING CUSTOM COLDFUSION TAGS AS DHTML WRAPPERS

Turn your
complex
cross-browser
DHTML
into clean
custom tags

BY TIM BUNTEL

On the one hand DHTML – the combination of HTML, stylesheets and JavaScript – has given us many new options to create low-bandwidth effects to enhance Web pages.

And on the other ColdFusion has given us the power to bring database contents to the Web. What's best for us, then, if we want interactive, site-enhancing effects that are data-driven? A combination of the two, of course! But it's never quite that simple....

This is especially true since DHTML has been implemented differently in Netscape and Microsoft browsers. Pages are often cluttered with a combination of

ColdFusion output and lines of script that are hard to debug, modify and reuse. The good news is that ColdFusion's custom tag creation abilities allow us to transform much of the complex client-side code into clean and reusable tags.

Ideal Marriage

To show how to create a ColdFusion custom tag DHTML wrapper, I'm going to focus on the example of our old friend, the ever-popular news-scroller widget.

The concept is simple: we want to display brief news headlines hyperlinked to full articles in a compact space on the page. This type of widget is ideally suited to a marriage between DHTML and Cold-

Fusion. A database holds information on what headlines to display and their links, and when to show them. ColdFusion is great at retrieving the items from the database and bringing them to the browser, but we need DHTML to control the showing and hiding of each one so as to create the "scrolling" effect.

The steps we take to turn this particular DHTML effect into a custom tag can be used with many other effects as well – all that's required is some careful planning.

There are two primary elements of the scroller:

- **CSS layers**, one for each news item, layered atop each other at the same window position

- **Script** to control the timed showing and hiding of each layer

The four steps to build the scroller are:

1. *Query the database for items to display.*
2. *Create a layer for each item.*
3. *Write a JavaScript function that will show and hide a given layer.*
4. *Write a timer with JavaScript to call the show-and-hide function every few seconds.*

Much of this is the same regardless of the implementation of the scroller, and that's our key to reusability. Any code that remains the same from one implementation to the next can be wrapped up. The tag that will ultimately be placed on our page needs to be concerned only with the variable elements. The headlines and their accompanying URLs, the position and size of the scroller, display elements such as font face and colors – everything else can be wrapped up in our tag.

As with other DHTML effects, the key is to separate the unique elements from the ones that are repeated. A navigation widget has unique pages and URLs but uses the same code to display them each time. Popup tool tips consist of different text each time but are always revealed with the same script. The unique elements will be parameters of the custom tag and the repeated pieces get wrapped up.

For our news-scroller, a syntax for the custom tag becomes clear – a main tag instantiating the scroller and child tags for each item. Something like:

```
<cf_NewsScroll with style and positioning
parameters>
<cf_NewsItem TheItem="..." TheURL="...">
<cf_NewsItem TheItem="..." TheURL="...">
(...)
</cf_NewsScroll>
```

Each time the page loads we want the ticker items built from the results of a database query. This is the key to making our dynamic HTML really dynamic. Thus:

```
<cfquery name="Headlines" datasource="News-
Database">
  SELECT Headline, URL
  FROM NewsTable
</cfquery>

<cf_NewsScroll with style and positioning
parameters>
<cfoutput query="Headlines">
<cf_NewsItem TheItem="#Headline#"
TheURL="#URL#">
</cfoutput>
</cf_NewsScroll>
```

That is to say, one child tag for each item returned from the database. For the ColdFusion developer who'll use this tag, the work is done (see Listing 3). With this simple syntax we can pass all the information necessary to construct the DHTML effect to another file for the dirty work.

The Dirty Work

The way all the elements come together to create our DHTML effect is a combination of ColdFusion custom tag development and cross-browser DHTML development. Either topic on its own would provide the basis for a lengthy article, so I won't deal with the specifics here except to show how to remove the code from the developer as we look into how to wrap up the news-scroller into a custom tag.

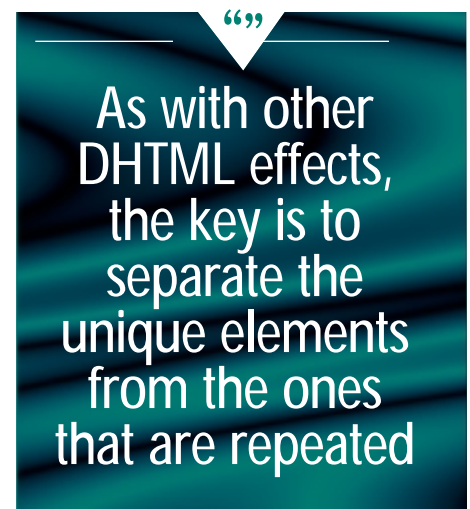
We don't need complex parent-child architected effects to justify creating wrappers. Simple, frequently used JavaScript snippets could be wrapped up into a tag taking only one or two variable parameters. Again, the idea is to hide the repetitive code.

Getting back to the example of our scroller, this does require a slightly more complex architecture. Since we never know in advance how many headlines will be displayed at runtime, the tag for this wrapper is really a tag family. There's a variable number of child tags, one for each headline and one main tag that pulls them all together and writes the layers and scripts. We'll need a separate cfm file for each: a parent tag called NewsScroll.cfm and an item template called NewsItem.cfm. In the calling page they'll be invoked with the syntax we used above:

```
<CF_NewsScroll >
<CF_NewsItem>
<CF_NewsItem>
(...)
</CF_NewsScroll>
```

The item template NewsItem.cfm consists of a single <CFASSOCIATE> tag (see Listing 1). Its sole purpose is to make its values – in our case each individual headline and its URL – visible to the parent tag via the AssocAttrs array. The length of this array will be crucial to the wrapper since it specifies the number of news items to be displayed.

The parent tag NewsScroll.cfm is where the real work is done. Since there's a start and end tag used on our calling page, NewsScroll.cfm will have two periods of activity identified as ExecutionMode. The Start mode is when the tag is first called and the End mode is when the close tag is called. We can't really build the DHTML until we've read in all of the



headlines, so most of the processing will be in the end ExecutionMode.

```
<CFIF ThisTag.ExecutionMode IS "END">
  <!--Do the work here...-->
</CFIF>
```

Parameters of CF_NewsScroll on the calling page can describe where and how to display the scroller: fonts, colors, screen coordinates. In NewsScroll.cfm we read in these parameters with the Attributes type and use defaults if no values are provided for the optional items.

Required attribute...

```
<CFSET TheHeight=#Attributes.Height#>
```

Optional attribute...

```
<CFIF ParameterExists(Attributes.FontFace)>
  <CFSET TheFontFace=#Attributes.FontFace#>
</cfif>
<CFSET TheFontFace="Arial, Helvetica">
</CFIF>
```

Now that we know where the scroller will be and how it will look, we're ready to begin creating the containers for each news item. Much of this work is done once for each news item, so start by setting a variable to this number. Again, the number of items is equal to the length of the AssocAttrs array.

```
<CFSET NumberOfNewsItems =
#ArrayLen(ThisTag.AssocAttrs)#>
```

Since later we'll be creating for... loops that start with zero, let's set a variable that's one less than the number of news items. (You'll see where we use this shortly!)

```
<CFSET ForNumber = NumberOfNewsItems - 1>
```

Each new item will be in its own container, here an HTML <div> tag, with the

container's properties defined by a style. We dynamically create each DIV and style with JavaScript's document.write command. See the complete code in Listing 2.

A style for each DIV...

```
document.write(' <STYLE TYPE="text/css">\n')
<!-- Loop through items -->
<CFLLOOP INDEX="i" FROM="0" TO="#ForNumber#">
```

```
document.write(' #NewsScrollItem<cfoutput>#i#
</cfoutput>{
// Use additional document.write commands
for properties
document.write(' }\n')
</CFLLOOP>
document.write(' </STYLE>\n')
```

will make something looking like this on the calling page:

```
<STYLE TYPE="text/css">
#NewsScrollItem1{
property: value; }
#NewsScrollItem2{
property: value; }
</STYLE>
```

And each DIV itself...

```
<CFLLOOP INDEX="i" FROM="1" TO="#NumberOfNewsItems#">
document.write(' <DIV
id="NewsScrollItem<cfoutput>#NumOfLayer#</cfoutput>">')
//Write the HTML for the item and its link
here...
```

```
document.write(' </DIV>' )
</CFLLOOP>
```

will result in something like this:

```
<DIV id="NewsScrollItem1">
<a href="http://www.wherever.com">A Big
News Headline! </a>
</DIV>
```

At this point we have one hidden container for each news item. All that remains is to create some more script to show and hide the containers on a timer. For our effect to be cross-browser compatible, we need to do some syntactical workarounds. See Listing 2 for how it's done.

As I said before, a full look into cross-browser DHTML is beyond the scope of this article, so try to take the code lightly. (If you're not familiar with DHTML, one look at this block alone ought to be sufficient argument for creating these wrappers as it's precisely this complexity that we're hiding!)

The last element is to set a timer for the show/hide function. The time between rotations is specified in milliseconds; in Listing 2 the value is 3,000 (three seconds). This too could well be a parameter of the tag to allow the developer to control the speed of the scroll.

If we didn't create this scroller as a custom tag we'd need to write this script on every page. By wrapping it up, our effect is achieved with the simple syntax we specified above:

```
<CF_NewScroll with style and positioning
```



```
parameters>
<CF_NewScroll TheItem="..." TheURL="...">
<CF_NewScroll TheItem="..." TheURL="...">
(...)
</CF_NewScroll>
```

The complexities are concealed, and by adding the tag to your ColdFusion custom tags directory you enable the scroller to be called from within any of your ColdFusion pages.

ABOUT THE AUTHOR

Tim Buntel is a senior Web developer for UNICOM Inc., an Allaire Premier Level Partner and technology consulting company offering Internet, data and LAN/WAN solutions to business, education and corporate clients throughout New England.

TBUNTEL@UNICOM-INC.COM

LISTING 1: The Child Tag, NewsItem.cfm

```
<!-- Associate this tag with CF_Ticker -->
<cfassociate BASETAG="cf_NewScroll">
```

LISTING 2: The Parent Tag, NewsScroll.cfm

```
<!-- Only process in END execution mode -->
<CFIF ThisTag.ExecutionMode IS "END">
```

```
<!-- Create Ticker placement variables (required attributes) -->
```

```
<CFSET NewsX=#Attributes.NewsX#>
<CFSET NewsY=#Attributes.NewsY#>
<CFSET NewsW=#Attributes.NewsW#>
<CFSET NewsH=#Attributes.NewsH#>
```

```
<cfif ParameterExists(Attributes.FontFace)>
<CFSET TheFontFace=#Attributes.FontFace#>
<cfelse>
<CFSET TheFontFace="Arial, Helvetica">
</cfif>
<cfif ParameterExists(Attributes.FontColor)>
<CFSET TheFontColor=#Attributes.FontColor#>
<cfelse>
<CFSET TheFontColor="black">
</cfif>
<cfif ParameterExists(Attributes.FontSize)>
<CFSET TheFontSize=#Attributes.FontSize#>
```

```
<cfelse>
<CFSET TheFontSize="3">
</cfif>
<cfif ParameterExists(Attributes.BGColor)>
<CFSET TheBGColor=#Attributes.BGColor#>
<cfelse>
<CFSET TheBGColor="CCCCC">
</cfif>
<cfif ParameterExists(Attributes.BorderColor)>
<CFSET TheBorderColor=#Attributes.BorderColor#>
<cfelse>
<CFSET TheBorderColor=TheBGColor>
</cfif>
```

```
<cfset NumberOfNewsItems = #ArrayLen(ThisTag.AssocAttrs)#>
<cfset ForNumber = NumberOfNewsItems - 1>
```

```
<script language="JavaScript">
<!--
```

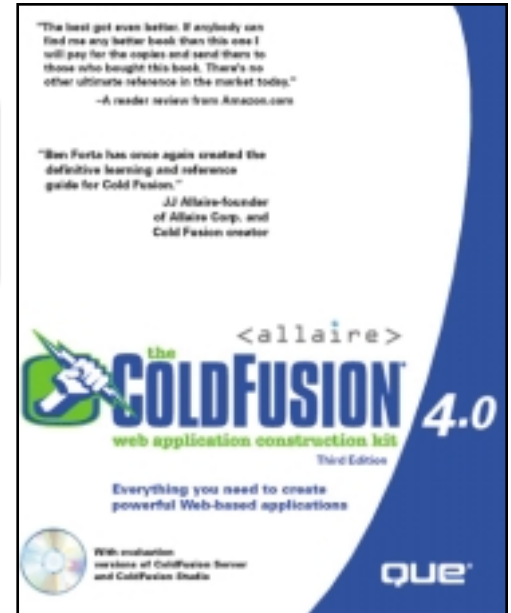
```
// BrowserCheck
ns = (document.layers)? true: false
ie = (document.all)? true: false
```


A Beginner's Guide to ColdFusion

COLDFUSION FORMS

Part 1

Using and creating forms, and processing form submission



FROM THE BOOK
BY BEN FORTA

Using Forms

In the previous two articles (*CFDJ*, Vol. 2, issues 2 and 3) on "ColdFusion Basics," you learned how to create ColdFusion templates that dynamically display data retrieved from ODBC data sources. The A2Z Employees table has just 10 rows, so the data fits easily within a Web browser window.

What do you do if you have hundreds or thousands of rows? Displaying all that data in one long list is impractical. Scrolling through lists of names to find the one you want just doesn't work well. The solution is to enable users to search for names by specifying what they are looking for. You can allow them to enter a first name, a last name, or part of a name, and then you can display only the employee records that meet the search criteria.

To accomplish this solution, you need to do two things. First, you need to create your search form using the HTML form tags. Second, you need to create a template that builds SQL SELECT statements dynamically based on the data collected and submitted by the form.

Creating Forms

Before you can create a search form, you need to learn how ColdFusion interacts with HTML forms. Listing 1 contains the code for a sample form that prompts for a first and last name. Create this template and save it in the C:\A2Z\SCRIPTS\12 directory as FORM1.CFM.

In your browser, type `http://your-server.com/a2z/12/FORMS1.cfm` to display the form as shown in Figure 1.

This form is simple, with just two data entry fields and a submit button, but it clearly demonstrates how forms are used to submit data to ColdFusion.

Using HTML Form Tags

You create HTML forms by using the `<FORM>` tag. `<FORM>` usually takes two parameters passed as tag

attributes. The `ACTION` attribute specifies the name of the script or program that the Web server should execute in response to the form's submission. To submit a form to ColdFusion, you simply specify the name of the ColdFusion template that will process the form. The following example specifies that the template `forms2.cfm` should process the submitted form:

```
ACTION="forms2.cfm"
```

The `METHOD` attribute specifies how data is sent back to the Web server. All ColdFusion forms must be submitted as type POST.

Caution: The default submission type is not POST; it is usually GET. If you omit the `METHOD="POST"` attribute from your form tag, you run the risk of losing form data – particularly in long forms or forms with `TEXTAREA` controls.

Your form has only two data entry fields: `<INPUT TYPE="text" NAME="FirstName">` and `<INPUT TYPE="text" NAME="LastName">`. Both create simple text fields. The `NAME` attribute in the `<INPUT>` tag specifies

This article has been adapted from the first part of Chapter 12 of *ColdFusion 4 Web Application Construction Kit* by Ben Forta. Published by permission of Macmillan Publishers Ltd. and the author. Chapter 11 appeared in two parts in the February and March issues of *ColdFusion Developer's Journal*. Part 2 of this article and Chapter 13 will appear in forthcoming issues.

the name of the field, and ColdFusion uses this name to refer to the field when it is processed.

Each form in a field is typically given a unique name. If two fields have the same name, both sets of values are returned to be processed and are separated by a comma. You usually want to be able to validate and manipulate each field individually, so each field should have its own name. The notable exceptions are the check box and radio button input types, which are described later in this chapter.

The last item in the form is an INPUT submit type. The submit input type creates a button that, when clicked, submits the form contents to the Web server for processing. Almost every form has a submit button (or a graphic image that acts like a submit button). The VALUE attribute specifies the text to display within the button, so `<INPUT TYPE="submit" VALUE="Process">` creates a submit button with the text Process in it.

Understanding ColdFusion Error Messages

If you enter your name into the fields and submit the form right now, you receive a ColdFusion error message like the one shown in Figure 2. This error says that template `C:\A2Z\SCRIPTS\12\FORMS2.CFM` cannot be found.

This error message, of course, is perfectly valid. You submitted a form to be passed to ColdFusion and processed with a template, but you have not created that template yet. Your next task, then, is to create a template to process the form submission.

Processing Form Submissions

To demonstrate how to process returned forms, you need to create a simple template that echoes the name you enter. The template is shown in Listing 2.

Processing Text Submissions

By now the CFOUTPUT tag should be familiar to you; you use it to mark a block of code that ColdFusion should parse and process. The line `Hello #FirstName# #LastName#` is processed by ColdFusion. `#FirstName#` is replaced with the value you entered into the `FirstName` field, and `#LastName#` is replaced with the value in the `LastName` field in Form 1. See *CFDJ*, Vol. 2, issue 3 for a detailed discussion of the ColdFusion CFOUTPUT tag.

Create a template called `FORMS2.CFM` that contains the code in Listing 2 and save it in the `C:\A2Z\SCRIPTS\12` directory. Resubmit your name by clicking the form's submit button once again. This time you should see a browser display similar to the one shown in Figure 3. Whatever name you enter into the First Name and Last Name fields in Form 1 is displayed.

Processing Check Boxes and Option Buttons

Other input types that you will frequently use are check boxes and option buttons. Check boxes are used to select options that have one of two states: on or off, yes or no, and true or false. To ask a user if he wants to be notified of book availability via e-mail, for example, you would create a check box field. If the user selects the box, his name is added to the mailing list; if the user does not select the box, his name is not added.

Option buttons are used to select one of at least two mutually exclusive options. You can implement a field prompting for payment type with options such as Cash, Check, Credit card, or P.O., for example, as an option button field.

The code example in Listing 3 creates a form that uses both option buttons and check box fields.

Figure 4 shows how this form appears in your browser.

Before you create `FORMS4.CFM` to process this form, you should note a couple of important points. First look at the four lines of code that make up the Payment Type option button selection. Each one contains the exact same NAME attribute – `NAME="PaymentType"`. The four input fields clearly have the same name, so your browser knows that they are part of the same field. If each option button has a separate name, the browser

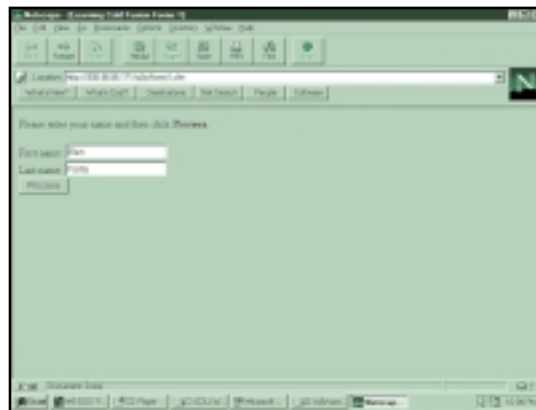


FIGURE 1: You can use HTML forms to collect data to be submitted to ColdFusion.



FIGURE 2: ColdFusion returns an error message when it cannot process your request.

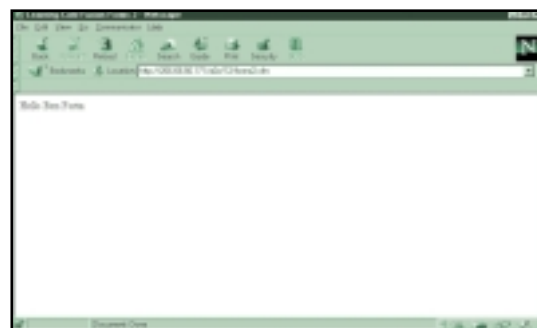


FIGURE 3: Submitted form fields may be displayed simply by referring to the field name.



FIGURE 4: You can use input types of option buttons and check boxes to facilitate the selection of options.

TIP

When you're using an INPUT type of submit, you should always specify button text by using the VALUE attribute. If you don't, the default text Submit Query (or something similar) is displayed, and this text is likely to confuse your users.

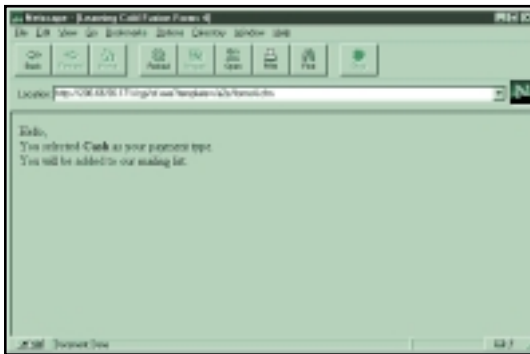


FIGURE 5: You can use ColdFusion templates to process user-selected options.

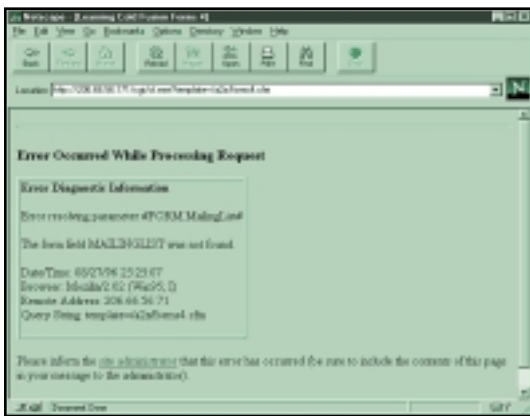


FIGURE 6: Option buttons or check boxes that are submitted with no value generate a ColdFusion error.

does not know that these buttons are mutually exclusive and thus allows the selection of more than one button.

Another important point is that, unlike INPUT type text, option buttons have no associated text or data entry area. Therefore, you must use the VALUE attribute in order for the browser to associate a particular value with each option button. The code VALUE="Cash" instructs the browser to return the value Cash in the PaymentType field if that button is selected.

Now that you can accept option button and check box fields, you're ready to create a template to process them. Create a template called FORMS4.CFM in the C:\A2Z\SCRIPTS\12 directory using the template code in Listing 4.

The form processing code in Listing 4 displays the payment type you select. The field PaymentType is fully qualified with the FORM field type to prevent name collisions. (See "Specifying Field Types" [CFDJ, Vol. 2, Issue 3] for an explanation of name collisions and how to avoid them.)

When the check box is selected, the

value specified in the VALUE attribute is returned; in this case the value is Yes. If the VALUE attribute is omitted, the default value of on is returned.

Now load form FORMS3.CFM on your browser, select a payment option, and then select the check box. Click the Process button. Your browser display should look like the one shown in Figure 5.

That process worked exactly as intended, so now get ready to complicate things a little. Reload template FORMS3.CFM and submit it without selecting a payment type or with the MailingList field not selected. As a result, ColdFusion generates an error message, as shown in Figure 6. The field you do not select generates a "Form Field Not Found" error.

Check the code in Listing 3 to verify that the form fields do in fact exist. Why does ColdFusion report that the form field does not exist? That is one of the quirks of HTML forms. If you select a check box, the on value is submitted; nothing is submitted if you do not select the check box – not even an empty field. The same is true of option buttons: if you make no selection, the field is not submitted at all. (This behavior is the exact opposite of the text INPUT type, which returns empty fields as opposed to no field.)

How do you work around this limitation? You can choose from two solutions. The first: modify your form processing script to check which fields exist by using the #Is defined()# function and, if the field exists, process it.

The second: the simpler solution is to prevent the browser from omitting fields that are not selected. You can modify the option button field so that one option is preselected. The users cannot avoid making an option button selection, so they have to make a selection or use the preselected options. To preselect an option button, just add the attribute CHECKED to one of the buttons.

Check boxes are trickier because by their nature they have to be able to be turned off. Check boxes are used for on/off states, and, when the check box is off, there is no value to submit. The solution is to set a default value in the action template. This is done using a ColdFusion tag called <CFPARAM>, which allows you to create variables on-the-fly if they do not already exist. Look at this code:

```
<CFPARAM NAME="MailingList"
```

```
DEFAULT="No">
```

When ColdFusion encounters this line, it checks to see if a variable named MailingList exists. If yes, processing continues. If it does not exist, ColdFusion creates the variable and sets the value to whatever is specified in the DEFAULT attribute. The key here is that either way – whether the variable existed or not – the variable does exist once the <CFPARAM> tag is processed. It is therefore safe to refer to that variable further down the template code.

The updated form is shown in Listing 5. The first option button in the PaymentType field is modified to read <INPUT TYPE="radio" NAME="PaymentType" VALUE="Cash" CHECKED>. The CHECKED attribute ensures that a button is checked. The MailingList check box has a VALUE of Yes when it is checked, and the <CFPARAM> in the action page ensures that if MailingList is not checked, the value will automatically be set to No.

Create and save this template as C:\A2Z\SCRIPTS\12\FORMS5.CFM and then add the following code to the top of FORMS4.CFM:

```
<CFPARAM NAME="MailingList"
DEFAULT="No">
```

Try using it and experiment with the two fields. You'll find that this form is reliable and robust, and it does not generate ColdFusion error messages.

Processing List Boxes

Another field type you will frequently use is the list box. Using list boxes is an efficient way to enable users to select one or more options. If a list box is created to accept only a single selection, you can be guaranteed that a value is always returned. If you don't set one of the options to be

TIP

Pay close attention to what code you place within and without the <CFOUTPUT> block. Misplacing a <TR> or </TD> could result in a badly formatted HTML table, and some browsers may opt to not even display that table.

LISTING 1: FORMS1.CFM – HTML Forms Can Be Used to Collect and Submit Data to ColdFusion for Processing

```
<HTML>

<HEAD>
<TITLE>Learning ColdFusion Forms 1</TITLE>
</HEAD>

<BODY>

<FORM ACTION="forms2.cfm" METHOD="POST">

Please enter your name and then click <B>Process</B>.
<P>
First name:
<INPUT TYPE="text" NAME="FirstName">
<BR>
Last name:
<INPUT TYPE="text" NAME="LastName">
<BR>
<INPUT TYPE="submit" VALUE="Process">

</FORM>

</BODY>

</HTML>
```

LISTING 2: FORMS2.CFM – Processing Form Fields

```
<HTML>

<HEAD>
<TITLE>Learning ColdFusion Forms 2</TITLE>
</HEAD>

<BODY>

<CFOUTPUT>

Hello #FirstName# #LastName#
```

```
</CFOUTPUT>
```

```
</BODY>
```

```
</HTML>
```

LISTING 3: FORMS3.CFM – Using Option Buttons and Check Boxes

```
<HTML>

<HEAD>
<TITLE>Learning ColdFusion Forms 3</TITLE>
</HEAD>

<BODY>

<FORM ACTION="forms4.cfm" METHOD="POST">

Please fill in this form and then click <B>Process</B>.
<P>
Payment type: <BR>
<INPUT TYPE="radio" NAME="PaymentType" VALUE="Cash">Cash<BR>
<INPUT TYPE="radio" NAME="PaymentType" VALUE="Check">Check<BR>
<INPUT TYPE="radio" NAME="PaymentType" VALUE="Credit
card">Credit card<BR>
<INPUT TYPE="radio" NAME="PaymentType" VALUE="P. O.">P. O.
<P>
Would you like to be added to our mailing list?
<INPUT TYPE="checkbox" NAME="MailingList" VALUE="Yes">
<P>
<INPUT TYPE="submit" VALUE="Process">
```

```
</FORM>
```

```
</BODY>
```

```
</HTML>
```

LISTING 4: FORMS4.CFM – Processing Option Buttons and Check Boxes

```
<HTML>

<HEAD>
<TITLE>Learning ColdFusion Forms 4</TITLE>
```

Adhost

www.adhost.com

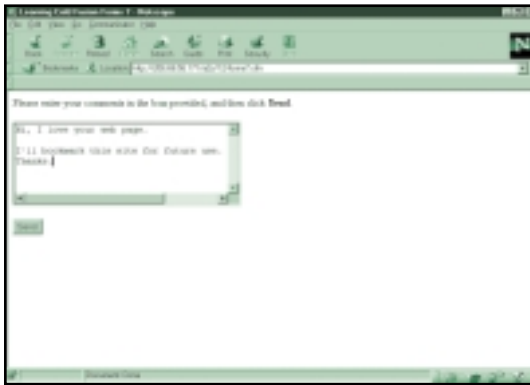


FIGURE 7: The HTML TEXTAREA field is a means by which you can accept free-form text input from users.

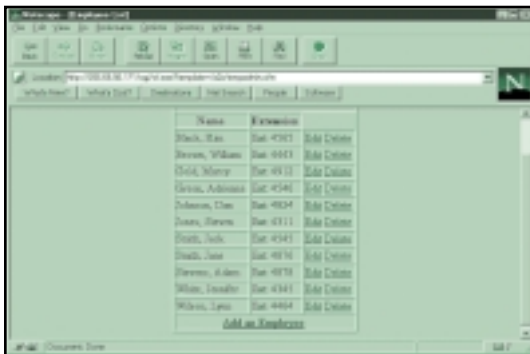


FIGURE 8: HTML tables are well suited for displaying query results.



FIGURE 9: Without ColdFusion output functions, TEXTAREA fields are not displayed with line breaks preserved.

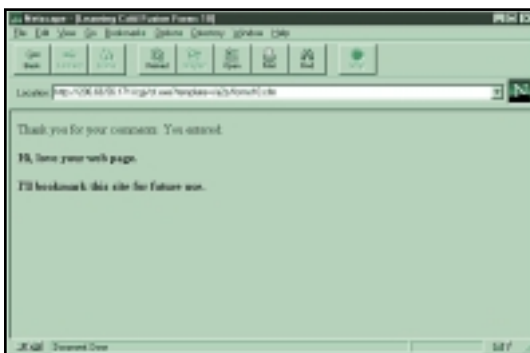


FIGURE 10: You should use the ColdFusion ParagraphFormat() function to display TEXTAREA fields with their line breaks preserved.

preselected, the first one in the list is selected. An option always has to be selected.

List boxes that allow multiple selections also allow no selections at all. If you use a multiple selection list box, you once again have to find a way to ensure that ColdFusion does not generate “Form Field Not Found” errors.

Listing 6 contains the same data entry form you just created but replaces the option buttons with a list box. Save this template as C:\A2Z\SCRIPTS\12\FORMS6.CFM and then test it with your browser.

For this particular form, the browser display shown in Figure 7 is probably a better user interface. The choice of whether to use option buttons or list boxes is yours, and no hard and fast rules exist as to when to use one versus the other. The following guidelines, however, may help you determine which to use:

- If you need to allow the selection of multiple items or of no items at all, use a list box.
- List boxes take up less screen space. With a list box, one hundred options take up no more precious real estate than a single option.
- Option buttons present all the options to the users without requiring mouse clicks.

Processing Text Areas

Text area fields are boxes in which the users can enter free-form text. When you create a text area field, you specify the number of rows and columns of screen space it should occupy. This area, however, does not restrict the amount of text that users can enter. The field scrolls both horizontally and vertically to enable the users to enter more text.

Listing 7 creates an HTML form with a text area field for user comments. The field's width is specified as a number of characters that can be typed on a single line; the height is the number of lines that are displayed without scrolling.

Listing 8 contains ColdFusion code that displays the contents of a TEXTAREA field.

Figure 8 shows the TEXTAREA field you created, and Figure 9 shows how ColdFusion displays the field.

Try entering line breaks (by pressing the Enter key) in the text field and then submit it. What happens to the line breaks? Line break characters are considered whitespace characters,

TIP

The TEXTAREA COLS attribute is specified as a number of characters that can fit on a single line.

This setting is dependent on the font in which the text is displayed, and the font is browser specific.

Make sure that you test any TEXTAREA fields in more than one browser because a field that fits nicely in one might not fit at all in another.

just like spaces, by your browser, and all whitespace is ignored by browsers. “WHITESPACE IS IGNORED” is displayed no differently than “WHITE-SPACE IS IGNORED.”

The only way to display line breaks is to replace the line break with an HTML paragraph tag: <P>. You therefore have to parse through the entire field text and insert <P> tags wherever needed. Fortunately, ColdFusion makes this task a simple one. The ColdFusion #ParagraphFormat()# function automatically replaces every double line break with a <P> tag. (Single line breaks are not replaced because ColdFusion has no way of knowing if the next line is a new paragraph or part of the previous one.)

The code in Listing 9 contains the same comments form as the one in Listing 7, with two differences. First, default field text is provided. Unlike other INPUT types, <TEXTAREA> default text is specified between <TEXTAREA> and </TEXTAREA> tags – not in a VALUE attribute.

Second, you use the WRAP attribute to wrap text entered into the field automatically. WRAP=“VIRTUAL” instructs the browser to wrap to the next line automatically, just as most word processors and editors do.

TIP

The ColdFusion Replace() and ReplaceList() functions may be used instead of ParagraphFormat() in order to have greater control over the paragraph formatting.

```

</HEAD>

<BODY>

<CFOUTPUT>

Hello, <BR>
You selected <B># PaymentType#</B> as your payment type. <BR>
<CFIF #MailingList# IS "Yes">
You will be added to our mailing list.
<CFELSE>
You will not be added to our mailing list.
</CFIF>
</CFOUTPUT>

</BODY>

</HTML>

```

LISTING 5: FORMS5.CFM – Using Hidden Fields to Set Default Form Values

```

<HTML>

<HEAD>
<TITLE>Learning ColdFusion Forms 5</TITLE>
</HEAD>

<BODY>

<FORM ACTION="forms4.cfm" METHOD="POST">

Please fill in this form and then click <B>Process</B>.
<P>
Payment type: <BR>
<INPUT TYPE="radio" NAME="PaymentType" VALUE="Cash"
CHECKED>Cash<BR>
<INPUT TYPE="radio" NAME="PaymentType" VALUE="Check">Check<BR>
<INPUT TYPE="radio" NAME="PaymentType" VALUE="Credit
card">Credit card<BR>
<INPUT TYPE="radio" NAME="PaymentType" VALUE="P.O.">P.O.
<P>
Would you like to be added to our mailing list?

```

```

<INPUT TYPE="checkbox" NAME="MailingList" VALUE="Yes">
<P>
<INPUT TYPE="submit" VALUE="Process">

```

```

</FORM>

```

```

</BODY>

```

```

</HTML>

```

LISTING 6: FORMS6.CFM – Using a SELECT List Box for User Options

```

<HTML>

<HEAD>
<TITLE>Learning ColdFusion Forms 6</TITLE>
</HEAD>

<BODY>

<FORM ACTION="forms4.cfm" METHOD="POST">

Please fill in this form and then click <B>Process</B>.
<P>
Payment type:
<SELECT NAME="PaymentType">
<OPTION>Cash
<OPTION>Check
<OPTION>Credit card
<OPTION>P.O.
</SELECT>
<P>
Would you like to be added to our mailing list?
<INPUT TYPE="checkbox" NAME="MailingList" VALUE="Yes">
<P>
<INPUT TYPE="submit" VALUE="Process">

</FORM>

</BODY>

</HTML>

```

Paperthin Inc.

www.paperthin.com



FIGURE 11: When you're using multiple submit buttons, you must assign a different value to each button.

Note: Many browsers do not support the TEXTAREA WRAP attribute. These browsers ignore the attribute and require the users to enter line breaks manually. Because the attribute is ignored when not supported, you can safely use this option when necessary; your forms do not become incompatible with older browsers.

Listing 10 shows the template to display the user-supplied comments.

The Comments field code is changed to `#ParagraphFormat(FORM.Comments)#`, ensuring that all line breaks are maintained and displayed correctly, as shown in Figure 10.

Processing Buttons


The HTML forms specification supports only two types of buttons. Almost all forms, including all the forms that you create in both parts of this article, have a submit button. Submit, as its name implies, instructs the browser to submit the form fields to a Web server.

The second supported button type is reset. Reset clears all form entries and restores default values if any existed. Any text entered into INPUT TYPE="text" or TEXTAREA fields is cleared, as are any check box, list box, and option button selections. Many forms have reset buttons, but you never need more than one.

On the other hand, you may want more than one submit button. For example, if you're using a form to

modify a record, you could have two submit buttons: one for Update and one for Delete. (Of course, you also could use two forms to accomplish this task.) If you create multiple submit buttons, you must name the button with the NAME attribute and make sure to assign a different VALUE attribute for each. The code in Listing 11 contains a reset button and two submit buttons.

The result of this code is shown in Figure 11.

When you name submit buttons, you treat them as any other form field. Listing 12 demonstrates how to determine which submit button was clicked. The code `<CFIF FORM.Operation IS "Update">` checks to see if the Update button was clicked, and `<CFELSEIF FORM.Operation IS "Delete">` checks to see if Delete was clicked, but only if Update was not clicked. 

BEN@FORTA.COM

LISTING 7: FORMS7.CFM – Using a Text Area Field

```
<HTML>

<HEAD>
<TITLE>Learning ColdFusion Forms 7</TITLE>
</HEAD>

<BODY>

<FORM ACTION="forms8.cfm" METHOD="POST">

Please enter your comments in the box provided, and then click
<B>Send</B>.
<P>
<TEXTAREA NAME="Comments" ROWS="6" COLS="40"></TEXTAREA>
<P>
<INPUT TYPE="submit" VALUE="Send">

</FORM>

</BODY>

</HTML>
```

LISTING 8: FORMS8.CFM – Processing Free-Form Text Area Fields

```
<HTML>

<HEAD>
<TITLE>Learning ColdFusion Forms 8</TITLE>
</HEAD>

<BODY>

<CFOUTPUT>

Thank you for your comments. You entered:

<P>

<B>#FORM.Comments#</B>

</CFOUTPUT>

</BODY>

</HTML>
```

LISTING 9: FORMS9.CFM – The HTML TEXTAREA Field with Wrapping Enabled

```
<HTML>

<HEAD>
<TITLE>Learning ColdFusion Forms 9</TITLE>
</HEAD>

<BODY>

<FORM ACTION="forms10.cfm" METHOD="POST">

Please enter your comments in the box provided, and then click
<B>Send</B>.
<P>
<TEXTAREA NAME="Comments" ROWS="6" COLS="40" WRAP="VIRTUAL">
Replace this text with your comments.
</TEXTAREA>
<P>
<INPUT TYPE="submit" VALUE="Send">

</FORM>

</BODY>

</HTML>
```

LISTING 10: FORMS10.CFM – Using the ParagraphFormat Function to Preserve Line Breaks

```
<HTML>

<HEAD>
<TITLE>Learning ColdFusion Forms 10</TITLE>
</HEAD>

<BODY>

<CFOUTPUT>

Thank you for your comments. You entered:

<P>

<B>#ParagraphFormat(FORM.Comments)#</B>

</CFOUTPUT>
```

</HTML>

<HTML>

<BODY>

<FORM ACTION="forms12.cfm" METHOD="POST">

$\langle P \rangle$

First name:

```
<INPUT TYPE="text" NAME="FirstName">
```


Last name:

<P>

```
<INPUT TYPE="reset" VALUE="Clear">
```

</FORM>

</BODY>

</HTML>

<HTML>

```
<HEAD>
<TITLE>Learning ColdFusion Forms 12</TITLE>
</HEAD>
```

<CFOUTPUT>

```
<CFIF FORM.Operation IS "Update">
You opted to <B>update</B> #FirstName# #LastName#
<CFELSEIF FORM.Operation IS "Delete">
You opted to <B>delete</B> #FirstName# #LastName#
</CFIF>
```

</CFOUTPUT>

</BODY>

</HTML>

DDDDDDDDDD

The code listing for
this article can also be located at
www.ColdFusionJournal.com

Infoboard
www.infoboard.com

SiteHosting.Net

www.SiteHosting.Net

JavaCon 2000

www.javacon2000.com

JavaCon 2000

www.javacon2000.com

Calling All Custom Tags

How and why to call them different ways

BY
CHARLES
AREHART



You probably know about custom tags, but are you aware of all the possibilities for controlling who can access them and how?

Perhaps you knew that a custom tag could be placed in the `cfusion\customtags` directory of the CF server to provide shared access by all users on the server, and you may even know that a tag placed in the same directory as the calling template will be found first (we'll call this a "local" custom tag).

But what if you want to share a custom tag – not among everyone on the server, but among only a handful of applications not even in the same directory? How do you solve *that* problem? This is also a real problem on hosted servers, where perhaps you're specifically precluded from adding to the shared `customtags` directory. What if you want to share a custom tag among several of your applications? Do you believe you have to place a copy in every directory you have on that server? That would negate almost entirely the benefits of easily reused code!

And have you ever wondered if you could literally prevent someone from executing a custom tag? (Still another benefit important to those

on shared servers.) Or did you know you could code your app such as to prevent someone from overriding a call to a shared custom tag with a local one? (Both these concerns would also be important in secured departmental or organizational environments.) Last, what if you wanted to call a custom tag whose name was in fact driven by a variable? All these things, and more, are possible.

There are at least six ways to control how and where a custom tag is called. In this month's **Journeyman ColdFusion** article, I'll expand upon these topics and discuss not only the language aspects of calling custom tags but also some advanced security aspects to control access to custom tags. We'll even throw in a couple of alternative ways of reusing code that are not quite custom tags but may be interesting to you.

For the Basics

In the February **CFDJ** (Vol. 2, issue 2), Ben Forta wrote a great article that included discussions of the basics of using and calling cus-

tom tags. He explained the important differences between using custom tags and `CFINCLUDE` and he alluded to a couple of advanced approaches to calling custom tags.

In case you're new to custom tags, let me simply restate that they're a great way to reuse code in ColdFusion applications. They're simply CFML files (C++ is supported as well) that perform some routine you may want to execute from within more than one template (.cfm file). The code in a custom tag is "called" using special CFML language syntax. I'll describe all those approaches as well as explain when and why you should use each approach.

Location, Location, Location

Given that the key benefit of custom tags is reuse of code, a very important aspect of using them is paying attention to just where the custom tag is stored so it can be shared among multiple programs. Of course, like all CFML files, it's to be placed on the ColdFusion server. But beyond that, there are important considerations that affect the choice of location as well as the language syntax used to call them.

Some folks are under the mistaken presumption that there are only one or perhaps two places where they may be stored. In fact, you can place a custom tag anywhere on the ColdFusion server. This has tremendous and often untapped benefits for sharing a custom tag among everyone on the server, just those in a particular application directory or those within a logical group of applications. That last choice is the one that many may not know.

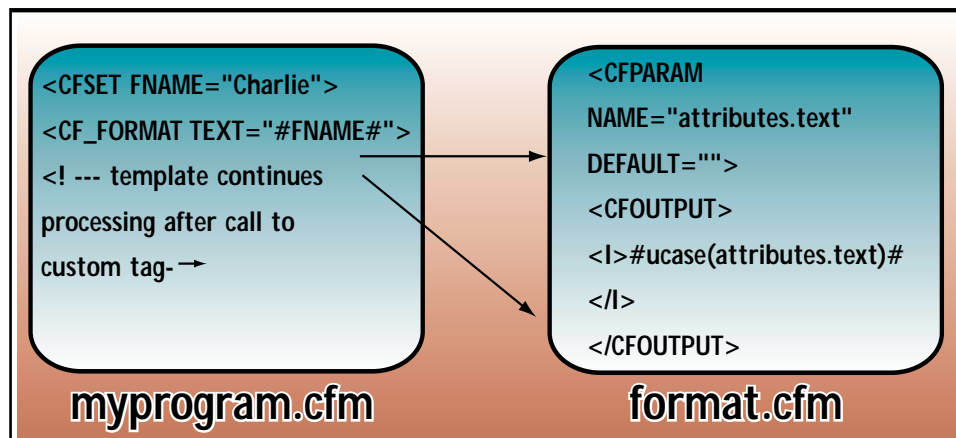


FIGURE 1: Example of a call to a very simple custom tag

Knock, Knock

The first matter to clarify concerns the many ways to call a custom tag. This is a quick reminder of the various approaches, with explanations of why you would use each approach. There are basically three approaches to calling them, with a couple of variations for spice.

CF_

Perhaps the most common approach that people will know is to refer to the custom tag by way of calling it with the “cf_” prefix. That is, if we had a custom tag called “format.cfm”, we could call it via:

```
<cf_format>
```

Notice that the file name of the tag follows the “cf_”, while the .cfm file extension is *not* used in the reference. When this code is encountered in a template, that first template is effectively placed on hold while the code in the custom tag is executed. The output of the custom tag, if any, is simply accumulated along with the output of the caller.

Again, just as a quick refresher, Figure 1 depicts a call to a very simple custom tag. What the custom tag does isn't really important (it simply causes text passed to it to be converted to uppercase and wrapped in HTML italics tags), but if you're new to using them, you can also see the means by which data can be passed to and used within a custom tag (providing parameter=value pairs after the call to the tag and use of the “attributes” prefix within the custom tag to refer to the passed parameters).

While this depicts the custom tag creating output that would simply be available as part of the output of the caller (when sent to the browser), it *is* possible – and sometimes highly desirable – to return data to the caller instead as a variable that can be processed entirely under the control of the caller. This involves using the “caller.” prefix in the custom tag to create such a variable, which would be available in the calling template – a subject beyond the scope of this article, but hopefully this is enough to get you started if you're new to it.

So that's a quick reminder of how custom tags work. The important point we were stressing was the question of how it was called, using the “cf_” syntax. Going back to that broader subject, the next issue to consider is where the custom tag will be found when processing occurs.

Where's Waldo?

We mentioned earlier that tags placed in the \cfusion\customtags directory where CF server is installed would be shared, that is, every appli-

cation on the server would have access to them. And this is true. A custom tag called using this syntax will find a custom tag of the given name, if it exists in the customtags directory.

But it will also look for the tag in any subdirectory of \cfusion\customtags, as well. That can be useful, though the more important benefit of such subdirectories will be explained in the next section.

We had also mentioned that a “local” custom tag could override the shared one, and this is true with

WinMill Software

www.winmill.com

this syntax, as well. That is, if the called custom tag is located in the same directory as the caller template, then the code in that local copy of the custom tag is what will be executed.

This can be very valuable when you're perhaps testing a new version of a custom tag. Rather than impact everyone on the server, you can have a testing directory in which your local copy is placed and only that code will see the local copy. Once it's ready for "prime time," it can be moved to the shared customtags directory.

These characteristics of how the custom tag can be found in either a local or shared location are handy, but there are times when either they will be inappropriate (you don't want to allow a local override), inaccurate (you want to execute a specific version of a custom tag in a particular subdirectory of \cfusion\customtags) or insufficient (you want to share a custom tag from templates in more than one directory, but you don't want to or can't place it in the shared customtags directory).

For these situations you may need to turn to yet another alternative for calling custom tags: the CFMODULE tag.

CFMODULE

CFMODULE is another way to call a custom tag that brings all the same benefits as the CF_ approach in terms of passing and returning data, as well as how processing in the caller is placed on hold and output of the custom tag is returned to the caller. The difference, besides the obvious change in tag syntax, is that CFMODULE allows much greater control over exactly where a custom tag is expected to be found. Sometimes you need that control, as we alluded to in the previous section. There are, in fact, two styles of CFMODULE: using the NAME or TEMPLATE attribute. Further, each of these two styles can be used two ways beyond their basic capability.

What's in a NAME?

We could also call our example "format.cfm" custom tag, above, using the following syntax:

```
<CFMODULE NAME="format">
```

Note that we still refer only to the *name* of the custom tag, not the ".cfm" extension.

And we could also pass any expected attributes to the custom tag by simply listing them after the call, just as we did with the "cf_" approach. Continuing from our example above, we could call it as:

```
<CFMODULE NAME="format"
TEXT="#fname#">
```

On the surface it may seem as though there's no difference, even that it may take a little more typing to achieve the apparently same result. But there's more here than meets the eye.

A very important aspect of the CFMODULE NAME= approach is that it will *not* look first in the local directory – that is, the directory where the calling template is located – for the custom tag. It will only look for it in the shared customtags directory. That could seem to be a limitation, but it's actually a benefit if for some reason you want your code to be sure *not* to execute anything except the shared custom tag, perhaps for security reasons.

There's an interesting and possibly important twist to using the CFMODULE NAME= approach. Used as described, it will search for the tag in the shared customtags directory, and like the CF_ approach it will also look below that directory to any subdirectories under \cfusion\customtags. But one thing that's unique to this approach is that you can name the *exact* subdirectory of custom tags in which you mean to find the custom tag.

That is, if you prefix the name of the custom tag (within the NAME attribute) with the name of the subdirectory, then only that specific subdirectory's version of the custom tag will be executed. For example, if we had a "salesapps" subdirectory under custom tags, and we placed our format.cfm file in that directory, we could execute that specific version of the custom tag using the following:

```
<CFMODULE NAME="salesapps.format">
```


```
TEXT="#fname#">
```

Notice that all that's changed is the prefix of "salesapps." before the custom tag name. This now forces the template to execute not any local version, not even the version in \cfusion\customtags or any random subdirectory of it, but *only* format.cfm as it exists in the \cfusion\customtags\salesapps directory.

This can be a useful feature once you understand its purpose. You might use it to organize shared custom tags to designate certain directories for certain departments or application groups (just to make a large number of tags more manageable and keep control under the right groups), or possibly to designate versions in a life cycle/change management paradigm with subdirectory names such as "dev", "qa" and "production".

(As an aside, there's no reason you couldn't use a global variable set in application.cfm to hold the name of the intended directory when executing in a particular mode, such as with <CFSET ctdir="dev">. Then you could use that variable in the call to the custom tag, as in <CFMODULE NAME="#ctdir#.format">. This is perfectly acceptable syntax.)

Finally, the NAME attribute can also specify not just a single subdirectory of the share customtags directory, but even "sub-subdirectories." In other words, if our format.cfm file were stored in a directory called \cfusion\customtags\salesapps\dev\, we could specify <CFMODULE NAME="salesapp.dev.format">. Notice that each subdirectory is simply specified as another "directoryname." prefix, in just the same order as the directories appear under \cfusion\customtags.

Next time, I'll look at what you can do if you *can't* place files in the customtags directory, by introducing you to the TEMPLATE= approach. 

Ed.Note: The following line under "So...Where to Search?" was left out of the bulleted list in last month's *Journeyman* column: "Search defusion, cfadvisor, cfdj and other sites."

ABOUT THE AUTHOR

Charles Arehart is an Allaire-certified instructor and developer working with Fig Leaf Software and is a frequent speaker at user groups throughout the country.

Career Opportunities

Fusedoc Redux

Keeping track of complex Web sites

BY
HAL
HELMS



Shortly after beginning to work with the Fusebox development methodology (see end of article for a brief list of Fusebox resources), I found that I wanted a standardized way of documenting code.

What I needed was something easy to learn and adopt, something that could provide important information without being obsessive in its detail. What I came up with is something I call *Fusedoc*, its name reflecting the fact that it was in large part inspired by Javadoc, the Java language's documentation standard. My column in the January 2000 issue of *CFDJ* (Vol. 2, issue 1) set out the underlying ideas and discussed implementation of a Fusedoc standard.

In my consultancy work I'm sometimes asked to implement the Fusebox development methodology at a company. Now this is tricky stuff! The old and largely discredited technique of management by fiat simply won't work. I view my job as helping developers see how Fusebox can (1) help them meet

the unholy deadlines they typically face and (2) increase their value to the company. Like the camel slipping his nose into the tent, I start with Fusedoc. I know it'll produce quick results. Once they overcome their initial skepticism, programmers tend quickly to discover for themselves the benefits of having concise, accurate documentation

The sequence of events usually goes something like this:

- **Phase 1:** "We don't need no stinkin' Fusedocs. I got enough work already just getting this code out the door before the next millennium."
- **Phase 2:** "Hmmm...Okay, maybe it's not as bad as I feared. I know customers and management like it and anything that gets them off my back is cool."
- **Phase 3:** "I had to go back and

make changes to Joe's code. I hate working with other people's code. But I was able to do it really fast because of the Fusedoc comments. I'm liking that. A lot."

- **Phase 4:** [Me]: "Now, these test harnesses are going to get thrown away eventually, so we can just skip the Fusedocs for them..."

[Them]: "Hold on, pal! If it gets written here, it gets documented with Fusedoc! You consultants..."

Once coders begin working with it, they begin to see new ways to build with Fusedoc. Experienced developers, for example, may architect large sections of code, defining all at once the fuses that will make up these sections. And since they're this far, why not fill in the Fusedoc comments at the same time? This produces something I call *fusestubs* – code files containing enough information for more junior coders to pick up and code.

What percentage of code that you write does someone else actually write? The greater your knowledge, the greater this percentage gets; you're far more valuable to the company if you have a system that lets you work at the level of your greatest expertise. Without such a system, senior coders will forever be creating add/edit/update forms.

While such a system obviously benefits both the company (faster development time, less cost) and the senior developers (more enjoyable work, better pay), it also benefits junior developers – providing them with the valuable experience of seeing how their code fits into the overall structure of an application

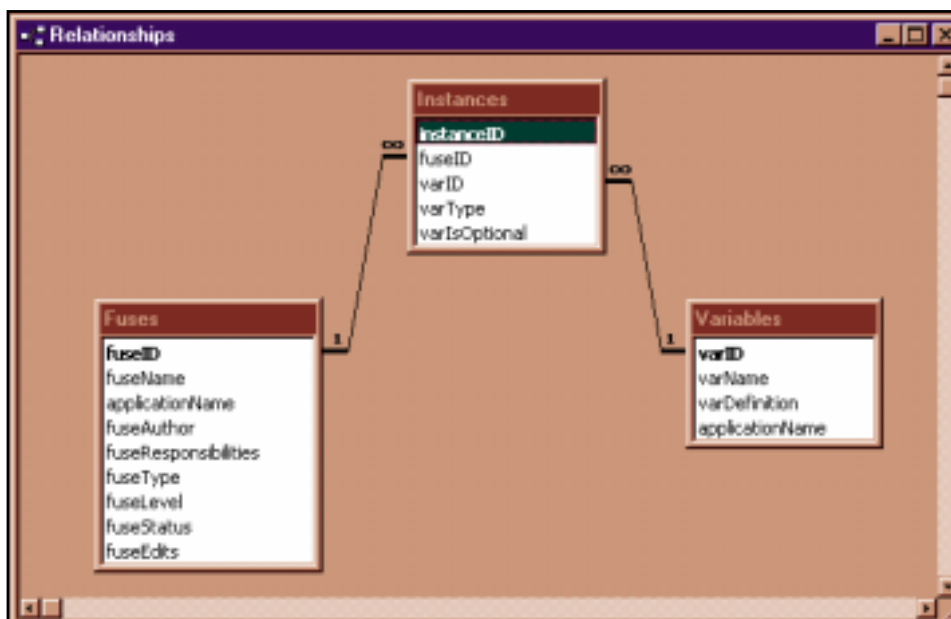


FIGURE 1: The relationships data schema

Career Opportunities

Career Opportunities

- Name of the fuse

47

ADVERTISING INDEX

ADVERTISER	URL	PH	PG
ABLE SOLUTIONS	WWW.ABLECOMMERCE.COM	360.253.4142	2
ADHOST	WWW.ADHOST.COM	888 ADHOST-1	31
BIZNIZ WEB	WWW.WEBPUBLISHINGTOOLS.COM	281.367.4016	50,52
CAREER OPPORTUNITIES		800.582.3089	51
CATOUZER	WWW.CATOUZER.COM	604.662.7551	55
COMPUTERJOBS.COM	WWW.COMPUTERJOBS.COM		3
DIGITAL NATION	WWW.DEDICATEDSERVER.COM	703.642.2800	4
EKTRON	WWW.EKTRON.COM	603.594.0249	9
ENHANCED TECHNOLOGIES	WWW.ENHTECH.COM	800.368.3249	39
ENTERACT	WWW.ENTERACT.COM	312.955.3000	23
EPRISE	WWW.EPRISE.COM	800.274.2814	17
INFOBOARD	WWW.INFOBOARD.COM	800.514.2297	50
INTELIANT	WWW.INTELIANT.COM	800.815.5541	21
INTERMEDIA.NET	WWW.INTERMEDIA.NET	650.424.9935	56
ISITE DESIGN	WWW.ISITEDESIGN.COM	888.269.9103	50
JAVACON 2000	WWW.JAVACON2000.COM		34-35
ON-LINE DATA SOLUTIONS	WWW.COOLFUSION.COM	516.737.4668	39
PAPERTHIN INC.	WWW.PAPERTHIN.COM	800.940-3087	33
RSW SOFTWARE	WWW.RSWSOFTWARE.COM	508.435.8000	15
SAISOFT	WWW.SAISOFTONLINE.COM	860.793.6681	52
SD 2000	WWW.SDEXPO.COM		19
SITEHOSTING.NET	WWW.SITEHOSTING.NET	888.463.6168	40
VIRTUALSCAPE	WWW.VIRTUALSCAPE.COM	212.460.8406	49
WATCHFIRE	WWW.WATCHFIRE.COM	613.599.3888	11
WINMILL SOFTWARE	WWW.WINMILL.COM	888.711.MILL	43
XMLDEVCON 2000	WWW.XMLDEVCON2000.COM		26-27

SYS-CON

www.sys-con.com

Able Solutions

Enter the realm of browsable store building and administration - from your browser. Build "your_site.com" with secure Merchant Credit Card Processing. Maintain inventory, add discounts and specials to keep your customers coming back. Increase sales with cross selling and membership pricing.

11700 NE 95th Street, Suite 100, Vancouver, WA
www.ablecommerce.com • 360 253-4142

Adhost Internet Advertising

Adhost provides complete web hosting solutions for over twelve hundred business clients. Small firms to multi-nationals, startups to long established companies - every business with which we do business receives the unparalleled level of service and range of products that has set Adhost Internet apart from the pack since 1995.

400 108th Avenue NE, Suite 700, Bellevue, WA 98004
www.adhost.com • (888) ADHOST-1

BiznizWEB
Internet Publishing
*Big Bucks? Heavy Web Expertise?
Operations Nightmare?
It doesn't have to be that way!*
www.webpublishingtools.com
local portals city guides calendars classifieds

DYNAMIC WEB PUBLISHING
recruitment directories
ad server polls

Catouzer

Catouzer develops web-based intranet and Customer Relationship Management software solutions. With Synergy 2.0, Catouzer continues its lead in providing secure web-based work environments. ColdFusion developers now have the most advanced framework to develop secure web-based projects.

www.catouzer.com • 604 662-7551

Computerjobs.com

ComputerJobs.com is the leading Internet-based job search company in its current markets. ComputerJobs.com is solely dedicated to helping computer and information technology ("IT") professionals find great jobs with companies in need of IT employees. We provide thousands of job listings for candidates seeking jobs in the IT field. We also provide companies who list jobs with us access to the resumes of the hottest IT professionals on the Web. Our user-friendly, proprietary Web site provides IT job seekers and hiring companies a convenient, effective way to connect.

3200 Windy Hill Road, Suite 700 West, Atlanta, GA 30339
www.computerjobs.com

digitalNATION - a VERIO company

digitalNATION, VERIO's Advanced Hosting Division, is the world's leading provider of dedicated server hosting, with over 1,650 servers online today. dN's superior connected network and service abilities have earned dN a solid reputation as a first-choice provider of dedicated server solutions (Sun, Windows NT, Linux and Cobalt). digitalNATION has been providing online and network services for over six years. One of the first ISPs to provide dedicated servers running Microsoft Windows NT, the dN staff has unparalleled experience in this industry.

5515 Cherokee Ave, Alexandria, VA 22312-2309
www.dedicatedserver.com • 703 642-2800

Ektron

Ektron supports the next-generation needs of e-businesses by providing dynamic Web infrastructure solution tools designed for use by nontechnical staff. Ektron's flagship offering, eContentManager, gives staff members across an organization the hands-on ability to make real-time additions and updates to Web content without requiring knowledge of a programming language -- while still allowing for centralized administrative control and security. With competitive advantages such as ease-of-integration and drag & drop everything, Ektron is looking to provide these empowering products to customers, resellers and integrators.

5 Northern Blvd., Suite 6, Amherst, NH 03031
www.ektron.com • 603-594-0249

Enhanced Technologies

Enhanced Technologies is a state of the art Web Consulting, Development, and Marketing provider built on the principles of ease of use, strength of service, and power of presentation. ETI is not just another firm offering Web development services. We are a professional Web Development company with many more services and capabilities than most of our competitors. We offer HTML (Hypertext)

design, Web graphics design, and CGI programming, but we also offer 3D graphics, inline, Java, and Macromedia Director animation.

6422 Grovedale Dr., Suite 301E, Alexandria, VA 22310
www.enhtech.com • 800-368-3249

EnterAct

EnterAct is the Business Services Group of 21st Century Telecom - Chicago's only single-source, facilities-based provider of bundled telecommunications. Combine this with EnterAct's excellent customer service and technical support and it's easy to see why EnterAct is the premier Internet Service Provider in Illinois.

407 S. Dearborn, 6th Floor, Chicago, IL 60605
www.enteract.com • 312-955-3000

Eprise Corporation

If your customers are looking for a content management solution, Eprise Participant Server FastStart Kit for Allaire ColdFusion developers can save you time and resources. Participant Server is a flexible-content management framework that enhances high-value business relationships through the delivery of timely, targeted, Web-based communications.

1671 Worcester Road, Framingham, MA 01701
www.eprise.com • 800 274-2814

Inteliant

Inteliant Corporation, a leading ColdFusion consulting firm, has an outstanding reputation for providing highly skilled developers for Internet, Intranet, Extranet, Software Development, or any ColdFusion application. Our national practice has emerged to meet the evolving needs of our clients by providing resources onsite or developing remotely. Our company provides the most cost effective service in the industry and we strive to add value to your projects by minimizing expenses whenever possible. Inteliant... "Delivering Intelligent Solutions"

1150 Hancock Street, Suite 4, Quincy, MA 02169
www.inteliant.com • 800-815-5541

Intermedia, Inc.

Our advanced virtual hosting packages (powered by Microsoft Windows NT and Internet Information Server 4.0) offer an environment supporting everything today's advanced Web developer or sophisticated client could ask for. Complete ODBC support is available on plans B and C. We support Microsoft Index Server on all hosting plans.

953 Industrial Avenue, Suite 121, Palo Alto, CA 94303
www.intermedia.net • 650 424-9935

ISITE Design

ISITE Design is a customer-focused New Media firm celebrating two years of serving the Portland, San Francisco and Los Angeles markets. Our diverse team provides a full range of services for our clients. From conceptualization and original art generation to database integration and online marketing, our team is poised to provide your organization with the highest level of customer service and results.

615 SW Broadway Ste. 200, Portland, OR 97205
www.isitedesign.com www.buyspectra.com • 888-269-9103

On-Line Data Solutions

CoolFusion.com is dedicated to providing unique and powerful add-on solutions for ColdFusion development and implementation. The site is hosted and maintained by On-Line Data Solutions, Inc. - a leader in ColdFusion integration. Our ColdFusion integration products line is called inFusion - a combination of "infuse" and ColdFusion. For information about our flagship product, inFusion Mail Server, we invite you to read the online information (where you can also download the latest beta) and join the inFusion Mail Server support list.

24 Elm Street, Centereach, NY 11720-1706
www.coolfusion.com • 516 737-4668

PaperThin, Inc.

PaperThin offers a 100% ColdFusion-based packaged solution empowering users with out-of-the box Web publishing and dynamic content management functionality for internets, intranets and extranets. Rich, browser and role-based tools allow users to easily create, update, schedule and personalize content without HTML knowledge, while eliminating the need for client software, technical training or complex scripting. Open and extensible, developers can quite readily integrate custom ColdFusion code and applications.

267 King Caesar Road, Duxbury, Massachusetts 02332
www.paperthin.com • 800-940-3087

RSW Software

RSW Software is a wholly owned subsidiary of Teradyne, Inc., and specializes in Web application testing software. Established with the goal of providing best-in-class testing products, RSW offers a suite of products called the e-TEST Suite, which automates the process of testing business-critical Internet and intranet applications.

44 Spring Street, Second Floor, Watertown, MA 02172
www.rswsoftware.com • 508 435-8000

SaiSoft

As a recognized Allaire, Microsoft and IBM Solutions Provider, SaiSoft's Strategic focus is to become the most definitive Internet Architect, by building long lasting e-business development partnerships. Our target markets are organizations seeking to establish a competitive and Interactive e-business presence.

With development operations in India & the UK, SaiSoft also undertakes off-shore consultancy projects where a 'four-step implementation' model is adopted to meet client needs satisfactorily. The software training services provided by SaiSoft to its clients are not wholly exclusive to existing clients, but employees from other organizations and individuals alike are also catered to.

446 East Street, Plainville, CT 06062
www.saisoftonline.com • 860-793-6681

Sitehosting.NET

Successful electronic commerce starts at SiteHosting.net: a division of Dynatek Infoworld, Inc., which provides total Web development services. We offer personal and efficient customer service with reliability at value prices. All our plans include access to SSL (Secure Socket Layer). We support ColdFusion, Active Server Pages, Real Audio/Video, Netshow Server, and more. Our hosting price starts at \$14.95/month.

13200 Crossroads Parkway North, Suite 360, City of Industry, CA 91746
www.sitehosting.net • 877 684-6784

Virtualscape

Why host with Virtualscape? Nobody else on the Internet understands what it takes to host ColdFusion like we do. Virtualscape is the leader in advanced Web site hosting. From Fortune 500 extranets to e-commerce sites and more, developers recognize our speed, stability, reliability and technical support.

215 Park Avenue South, Suite 1905, New York, NY 10003
www.virtualscape.com • 212 460-8406

Watchfire

Watchfire is the leading provider of software solutions that give organizations the power to ensure the quality and usability of their websites. Watchfire was recently voted one of the fastest growing Independent Software Vendors in North America (Information Week) and one of Canada's 25 "Up and Comers" (Financial Post). The Company currently sells into more than 60 countries and has over 50,000+ installed customers including over 50% of Fortune 500 companies and many of the largest e-commerce, government and education sites on the Web.

135 Michael Cowpland Dr, Suite 400, Kanata, Ontario K2M 2E9 Canada
www.watchfire.com • 613-599-3888

WinMill Software

WinMill Software is the premier resource for Internet and Intranet development, expert consulting, certified classroom training, Web-based education, e-commerce, development software, and technical support with expertise in site design and development, instructional design and delivery, application development, site hosting, training and site management. WinMill Software was founded on the principle of supporting the entire project lifecycle and our hallmark is a knowledge transfer process that maximizes the skills and intellect of your Internet development associates. We will work with your team to build a stable system architecture.

420 Lexington Avenue, Suite 307, New York, NY 10170
www.winmill.com • 888-711-MILL

On-Line Data Solutions Releases iMS/FusionMail 1.1

(Centereach, NY) – On-Line Data Solutions, Inc., announces iFusion Mail Server version 1.1 – a free upgrade from version 1.0. Building on the iMS 1.0 platform, iMS 1.1 now includes flexible outgoing mail priority, MX caching, specific IP binding, ESMTP support, RCPT cache and iMS Test Suite.
www.coolfusion.com/iMS.htm

Allaire and Productivity Point Partner on Training

(Cambridge, MA) – Allaire Corporation and Productivity Point International have announced a distribution agreement to offer Allaire-authorized training on ColdFusion and HomeSite. Under the terms of this agreement, Productivity Point will provide training on Allaire

products in more than 100 cities across North America.

Training is being rolled out between January and April. Allaire will continue to offer training directly as well as through the existing network of Allaire-authorized training centers. www.allaire.com

Sitech Launches SiteMaker Development 2.0

(Birmingham, AL) – Sitech Software has announced the availability of SiteMaker Development 2.0, a Web development and management tool designed specifically for professional Web developers.

Key features and benefits include open source architecture, centralized site manage-

ment and preprogrammed modules.

www.sitechsoftware.com

Catouzer Launches Synergy 2.0

(Vancouver, BC) – Catouzer has released Synergy 2.0, the latest version of their flagship



product. Some of the new features

developed for 2.0 include support of LDAP and NT Domain directories, which allows users to be registered into the Synergy system using LDAP or NT usernames and passwords, enhanced reporting features that provide statistics on the usage patterns within Synergy, and improved role and group management features.

www.catouzer.com

eye media Eyes Virtual Auctioneer Patient

(Dallas, TX) – eye media, inc., recently filed with the U.S. Patent & Trademark Office to patent certain aspects of its Virtual Auctioneer product, which offers a packaged solution to support a variety of e-commerce transactions, including Internet auctions, exchanges and online stores.



Virtual Auctioneer has been licensed to a broad range of companies for use in such diverse net market auctions as the purchase and sale of oil and gas properties, retail products, online music, overstocked inventories, and other vertical marketing and B2B activities.
www.eyemedia.net

SaiSoft

www.saisoftonline.com

Online Data Solutions

www.coolfusion.com

Catouzer

www.catouzer.com

Intermedia.net

www.Intermedia.net.com